1 Introdução

Como explicado nos trabalhos práticos anteriores, 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 *SaudeGes*, centrado no apoio à gestão de atividades relacionadas com a saúde e bem-estar.

Com o sucesso da versão 1.0 do *SaudeGes* decidiu-se avançar para a versão 1.5 que tem como objetivo permitir a utilização dos serviços do sistema a partir de diversos dispositivos com ligações à *internet*.

Mais precisamente, nesta nova 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*.

Dada a urgência em lançar esta nova versão para o mercado (que é como quem diz, dado que o semestre se aproxima do fim rapidamente), 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 um utilizador realizar os casos de uso *Criar Atividade* e *Definir Novo Horário de Atividade Regular*
- o cliente *web* deve permitir um utilizador realizar, através de um navegador, os casos de uso *Comprar Participação Mensal em Atividade Regular* e *Agendar Atividade Ocasional*

2 Que devemos fazer?

Neste trabalho pretende-se que, baseada na vossa versão 1.0 do *SaudeGes*, produzam uma nova versão preparada para ter o negócio a correr num servidor aplicacional e dois clientes remotos (uma aplicação web e uma aplicação desktop). Para isso devem:

- adaptar a camada de negócio da aplicação de forma a que esta possa servir pedidos concorrentes e 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*.

O material de apoio fornecido 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 *SaudeGes*, o projeto 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:

- **saudges**, que agrega a informação dos restantes (não têm de programar nada neste projeto);

- **saudges-ear**, que contém informação de como empacotar a aplicação empresarial Java a instalar no servidor (não têm de programar nada neste projeto);
- **saudeges-business**, que contém a camada de negócio organizada de acordo com o padrão *Domain Model* e *Data Mapper JPA*;
- **saudeges-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 (padrões *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;
- **saudeges-gui-client**, que é um cliente que oferece 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 (*saudeges-business*):
  - Substituir as classes que vos fornecemos pelas classes que desenvolveram na primeira parte do trabalho (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 tratam dos casos de uso e 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 e onde é necessário fazer gestão de concorrência, utilizando as primitivas oferecidas pelo JPA (levando em consideração os quatro casos de uso considerados nesta iteração).

- No projeto correspondente à apresentação concretizada pelo cliente web (*saudeges-web-client*):
  - 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 de proceder ao desenvolvimento das ações.
  - As vistas (definidas usando o JSP) têm de ser refeitas 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 concretizada pelo cliente desktop (*saudeges-gui-client*):
  - 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 desta versão da aplicação *SaudeGes* vai ser distribuída pelas máquinas dos utilizadores, um servidor Java EE, e um servidor de base de dados. A base de dados idealmente estará numa máquina diferente da do servidor aplicacional. Como anteriormente, devem utilizar a máquina disponível no endereço http://dbserver.alunos.di.fc.ul.pt/. Para efeitos de desenvolvimento, usam um servidor Java EE (Wildfly) disponível localmente (nos vossos computadores).

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?

Devem importar o projeto fornecido. Trata-se de um projeto *maven* com o código de uma versão do sistema *SaleSys* com características semelhantes às que se pretende neste trabalho para o *SaudeGes*. O projeto é semelhante ao que vos fornecemos no início do semestre para testarem a instalação das ferramentas que vos dissemos que íamos usar em CSS. Recordem a forma como 1) se coloca a aplicação a correr no servidor aplicacional *Wildfly*, 2) se usa a aplicação web através de um *browser* e 3) se corre o cliente *desktop*.

Um elemento do grupo de trabalho deverá criar um repositório git com o nome css2021_projeto2_XXX, em que XXX corresponde ao número do grupo, e definir a visibilidade do projeto como privada. Além de dar acesso aos colegas de grupo com o nível *Maintainer*, deve adicionar à lista de membros do projeto o utilizador css000 com o nível de *Reporter*.

Coloquem o projeto fornecido no repositório e ponham as mãos à obra, não esquecendo de começar por alterar os ficheiros que definem as ligações à base de dados, de forma a passarem a usar a área do vosso grupo.

4 Como e quando entregamos?

Identifiquem o *commit* de entrega com `git tag entrega` e coloquem essa identificação no servidor gitlab (`git push origin entrega`). O *deadline* para entrega é **28 de maio**.

O repositório deve conter:

1. um único documento PDF, chamado `Relatorio`, com as decisões mais importantes que tenham tomado em termos de desenho da aplicação, nomeadamente a forma como resolverem cada um dos aspetos listados na secção sobre o que têm a fazer (por exemplo, que EJBs definiram, que interfaces definiram para esses EJBs, que problemas de concorrência identificaram e como são estes evitados, etc). Devem ainda mencionar modificações relevantes que tenham realizado relativamente à entrega da 1ª parte.
2. O código fonte do vosso projeto (o que inclui o .pom).
3. Alguns *screenshots* que ilustrem a apresentação da aplicação em cada um dos casos de uso (à semelhança do que existe no exemplo fornecido).

Dicas para Problemas Frequentes

- Façam *reorganize imports* em todas as classes de todos os módulos do projeto de forma a não deixarem quaisquer referências a classes do *SaleSys* que vão causar erros mais tarde.
- Se obtiverem erros na instalação da aplicação **analisem os erros que aparecem na consola** para tentar perceber o que está a correr mal.
- Têm de ter a aplicação a correr no servidor para poder usar qualquer uma das aplicações cliente.
- 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.
- Se obtiverem erros na execução do cliente *desktop*, **analisem os erros que aparecem na consola**. Se houver erros relacionados como JavaFx olhem de novo para os requisitos de instalação fornecidos no início do semestre, nomeadamente para tudo o que diz respeito ao Java.