CONSTRUÇÃO DE SISTEMAS DE SOFTWARE

Relatório Entrega 5

Engenharia Informática – 2019/2020

Grupo 41

Diogo Nogueira, Nº 49435

André Monteiro, Nº 51718

Decisões Importantes no Desenho da Aplicação

Camada de Negócio

Para desenvolver o projeto da meta 5, começamos por adaptar a nossa lógica de negócio à nova versão da aplicação. Esta nova versão do projeto AulaGes possui dois projetos cliente para executar os casos de uso desenvolvidos nas metas anteriores (GUI e Web), por isso, com o intuito de transportar informação entre estes foram definidas interfaces remotas definitas com a anotação @Remote. Estas, que se inserem no package facade, possuem a assinatura dos métodos dos diferentes casos de uso e, como tal servem para os projetos cliente os poderem executar sem que a sua implementação seja exposta. Como em metas anteriores tínhamos definido as classes service LessonService, RegistrationService e FacilityService que permitiam a clientes simples executar os casos de uso, desta vez cada um destes services implementa uma classe Remote com a finalidade anteriormente referida.

Os services para esta meta foram adaptados para serem session beans. Todos eles não guardam estado, logo são session bean stateless e possuem a anotação @Stateless. Estes, para executar os métodos dos diferentes handlers, necessitam que as dependências destes últimos sejam injetadas através da anotação @EJB nos atributos de cada service. Se um atributo de uma classe e essa mesma estiverem em diferentes camadas ou subcamadas (neste último caso as classes service do package facade e os seus atributos handlers do package business), então terão de ser anotados com as anotações @EJB e @Stateless ou @Stateful, respectivamente.

Seguindo esta linha de pensamento, para cada classe handler foi-lhe anotada um session bean. Os handlers CreateLessonHandler, ActivateLessonHandler e ShowOccupationHandler pertencem aos session beans de sessão sem estado, contudo, o handler RegiserLessonHandler é um session bean que guarda estado (em atributos como currentRegistrationType, currentSportModality e currentUser), pois para executar as suas operações é necessário realizar várias validações e interações entre o utilizador.

Como a comunicação entre a base de dados já não é dependente de EntityManagers criados em cada classe handler, estas transações foram substituídas nas classes que representam os catálogos (também eles session beans stateless) com uma única instância de um objeto EntityManager devido à anotação @PersistenceContext. Cada método que persiste um objeto na base de dados é anotado com @Transactional(Transactional.TxType.REQUIRES_NEW) se o objeto era novo na base de dados ou apenas @Transactional.

De modo a transferir a informação necessária para mostrar ao utilizador e proteger a integridade dos dados enviados remotamente, foi utilizado o padrão Data Transfer Object. Para cada método de um handler com um tipo de objeto de retorno mais sofisticado foram criados diferentes objetos DTO serializáveis com atributos com tipos de dados mais simples, tais como:

ActiveLessonDTO, FacilityDTO, LessonDTO, OccupationDTO, RegistrationDTO e SportModalityDTO.

A nível de concorrência o nosso grupo definiu atributos com a anotação @Version em duas classes, sendo elas Registration e Lesson, por serem as responsáveis por criar os diferentes objetos e relações nos diferentes casos de uso.

Cliente Web

No desenvolvimento do cliente web utilizamos as interfaces anotadas com @Remote para efetuar os casos de uso Inscrever em Aula e Visualizar Ocupação. De maneira a dar utilidade a essas interfaces foi utilizado o padrão Front Controller em que são utilizadas diferentes classes Action. A correspondência entre os endereços web e as classes Action é mantida no ficheiro app.properties que contem os URL's seguintes:

appRoot/action/inscricoes/inscreverEmAula = java:module/NewRegistrationAction

appRoot/action/inscricoes/confirmarDados = java:module/ChooseSportAndRegistrationTypeAction

appRoot/action/inscricoes/finalizarInscricao = java:module/CompleteRegistrationAction

appRoot/action/ocupacoes/visualizarOcupacoes = java:module/NewShowOccupationAction

appRoot/action/ocupacoes/ocupacao = java:module/ShowOccupationAction

A visualização é mantida com ficheiros JSP que utilizam as diferentes Actions para executar as várias funcionalidades da aplicação.

Cliente GUI

No projeto de cliente GUI foram criados três ecrãs: menu principal, criar aula e ativar aula. O menu principal serve apenas para o utilizador escolher qual o caso de uso que quer executar. Cada ecrã responsável por um caso de uso e utiliza os session beans remotos para executar as suas funcionalidades através das classes controller, ActivateLessonController e CreateLessonController. Para estabelecer conexão entre o controler e a view (.fxml) foram implementadas as classes Model para cada objecto.