UNIVERSIDADE DO MINHO

Trabalho Prático 2

Mestrado Integrado em Engenharia Biomédica

Aplicações Distribuídas

(1ºSemestre/2014-2015)

A65078 Ana Sofia Quintas

A61777 Cármina Azevedo

A65037 Margarida Costa

Resumo

No âmbito da Curricular de Aplicações Distribuídas foi proposto a realização de um trabalho prático que consistiu no desenvolvimento de um aplicação Cliente/Servidor de gestão e para uma Unidade de Saúde. Este trabalho teve como principal objectivo, a implementação em JAVAEE da aplicação abaixo descrita.

A problematização deste exercício prático, passou pelo desenvolvimento de um Centro Hospitalar UMINHO com mais de 100 mil utentes, 5 mil médicos 7000 enfermeiros e 3000 funcionários onde se pretendeu criar um sistema integrado de gestão de consultas, ficha clínica, atos médicos, atos de enfermagem, gestão de farmácia hospitalar. Assim, o presente relatório resulta do proposta de resolução para o exercício proposto e serve como suporte teórico para o mesmo.

Índice

Introdução

O Java EE (Java Enterprise Edition) consiste numa série de especificações bem detalhadas, dando uma “receita” de como deve ser implementado um software e o que faz cada um desses serviços de infraestrutura. Desta forma há uma redução significativa do custo e da complexidade do desenvolvimento, implantação e gestão de aplicações de várias camadas centrados no servidor. O Java EE é construído sobre a plataforma Java SE e oferece um conjunto de APIs (interfaces de programação de aplicações) para desenvolvimento e execução de aplicações portáteis, robustas, escaláveis, confiáveis e seguras no lado do servidor.  Alguns dos componentes fundamentais para o funcionamento da aplicação Cliente/Servidor Java EE foram:

* EJB: Um dos objectivos principais do EJB consiste em fornecer um desenvolvimento simples e rápido de aplicações [Java](http://pt.wikipedia.org/wiki/Java_(linguagem_de_programa%C3%A7%C3%A3o)), com base em componentes distribuídos, transacionais, seguros e portáveis. Trata-se de uma arquitetura gerida do lado do servidor utilizada para encapsular a lógica corporativa de uma aplicação.
* JPA : uma estrutura que permite aos programadores gerir os dados utilizando o mapeamento objeto-relacional (ORM) em aplicações construídos na plataforma Java.

“Entity Bean”

Uma “entity bean” representa um dado persistente armazenado num banco de dados. Este é identificado pela sua chave primária e possui ainda as relações entre as outras entidades. A título de exemplo, pode-se verificar a classe Utente utilizada no desenvolvimento da aplicação representado na figura 1.

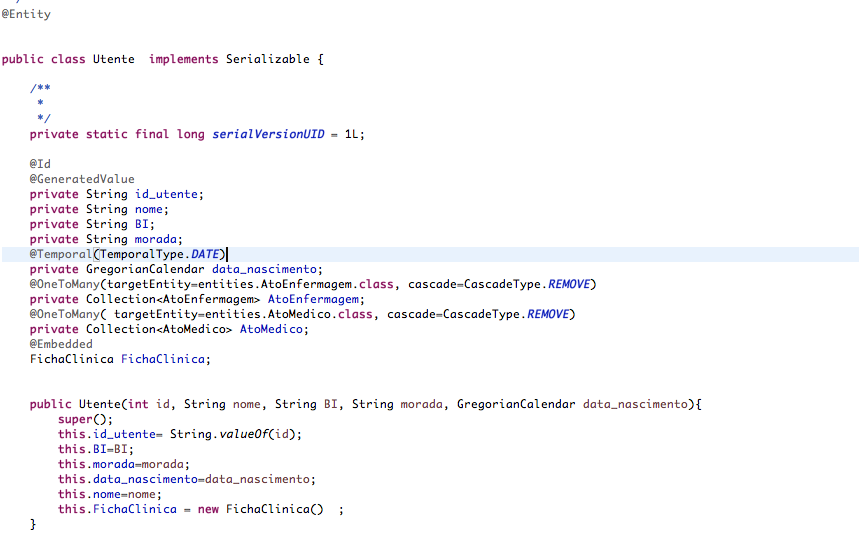


Figura 1 : Entity “Utente”

As classes Entity podem ser melhor especificadas quando adicionamos algumas anotações. Obrigatoriamente todas as entity do JPA precisa tem de possuir algumas anotações essências. São visíveis, na classe “utente” o recurso ao @Entity que identifica tratar-se de uma entidade, a utilização do *@id* que trata de identificar a chave primária e, ainda a utilização da anotação *@Temporal(TemporalType.Date)* que identifica o tipo de dados. Por outro lado é também através de anotações que é possível definir os relacionamentos entre outras entidades. É visível o recurso à anotação *@OneToMany,* que neste caso significa que um utente pode ter vários Atos Médicos, e da mesma forma vários Atos de Enfermagem.

É de notar que na Entidade “Ato Médico” verifica-se a anotação inversa *@ManytoOne.* Na anotaçãoo @OneToMany é ainda especificada nesta relação *fetch = FetchType.LAZY e cascate=cascade.Type.Remove.* O FetchType podemos definir a forma como serão trazidos os relacionamentos, neste caso são de forma “Lazy” significa que não serão carregados todos os dados relativos aos atos médicos e atos de enfermagem, deste modo prevê-se uma maior eficiência. Relativamente à ultima especiação da anotação, esta tem como função garantir que quando por exemplo um medico é apagado, os seu atos médicos sejam apagados com eles.

Por último, a anotação *@Embedded* define a classe “FichaClínica”, partiu-se do pressuposto que quando um utente fosse adicionado à base de dados este deve-se possuir obrigatoriamente uma Ficha de Cliente. Assim a anotação utilizada satisfaz este requisite, pois “embute” a classe Ficha Clinica e os seus atributos diretamente na entidade principal “Utente”. Desta forma não existe uma tabela exclusiva para a Ficha Clinica, e esta será registada na mesma tabela que o utente.

Apesar de não ser visível, as entidades são, ainda, constituídas pelos seus métodos de *get* e *set,* e ainda de adição e remoção das *collections.*

Sessions Beans

Os Sessions Beans são capazes de executar os vários serviços para os clientes. Assim, neste projeto foram definidos apenas dois Sessions Beans: o “HospitalManagement” e o “GestãoHospitalarManagement”. No primeiro estão definidas todas as operações da rotina de dia-adia que um hospital pode executar: criar utentes, enfermeiros, funcionários, marcar consultas, encomendar medicamentos, entre muitos outros. Do outro lado, a “GestãoHospitalarManagement” é destinada a um cliente que não está inserido no quotidiano das ações do hospital e apenas pretende “supervisionar” o seu funcionamento. Neste último session bean estão as funções de estatísticas como o Médico que mais prescreve, os medicamentos mais receitados, entre outros.

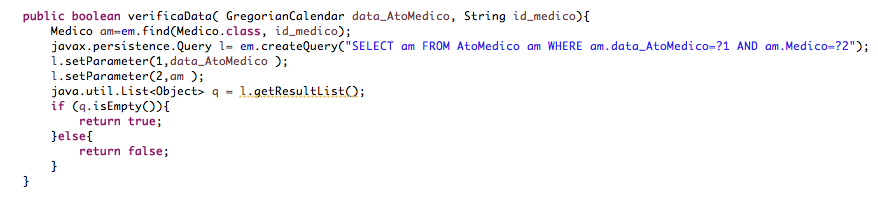
A JPA oferece suporte a consultas estáticas e dinâmicas (JPA, 2011). A API fornece uma linguagem própria de consulta, que é uma extensão da EJB QL (Enterprise Java Beans Query Language). Essa linguagem pode ser usada como uma alternativa ao SQL, que também é suportado. A criação de consultas é feita por meio do EntityManager, que fornece métodos específicos para instanciar consultas estáticas e dinâmicas . Neste trabalho apenas foram implementadas consultas do tipo dinâmico, tal como se pode verificar na figura 2. Neste tipo de consulta utilizou-se o método createQuery().  


Figura 2 -Função que verifica se já existe uma consulta para determinada hora.

Clientes

Neste projeto foram definidos três tipos de clientes, de acordo com os sessions beans implementados. Assim, em projetos separados e através da opção de adição de “*Aplication Client Project*” foram criados três clientes:

* AulaEntitiesClientAPP2
* GestãoHospitalar;
* ClienteEspecial;

O primeiro cliente comunica com a interface “HospitalManagementRemote”, e permite ao utilizador, o acesso aos métodos do quotidiano de um hospital, existentes na *Session Bean* “HospitalManagement”. Este cliente, é composto por uma *main*, suportada por várias outras classes de menus.

O segundo cliente está associado à interface remota “GestaoHospitalarManagementRemote” e permite ao seu utilizador aceder às funções de estatísticas, existentes na *Session Bean* “GestaoHospitalarManagement”.

Por último, o terceiro cliente utiliza a mesma interface que o primeiro, de modo a ter acesso aos métodos de criação das entidades, de forma a poder povoá-las.

Relacionamentos das “*Entities beans*”

Neste trabalho, a principal preocupação, foi um bom relacionamento entre as entidades de forma a que o todas as especificações do enunciado fossem respondidas. Assim, como se pode ver pelo figura 3, os relacionamentos entre as entidades são complexos, mas pode-se resumir ao seguinte:

* Um Médico pode executar Atos Médicos, que sejam previamente marcados. De um Ato Médico pode resultar uma prescrição.
* Um Utente poderá levantar a sua prescrição na farmácia, bastando para isso apresentar a sua identificação dentro da instituição.
* Um Enfermeiro pode executar Atos de Enfermagem, que sejam previamente marcados, durante o mesmo, caso haja utilização de material, este terá de ser registado.
* Os Atos Médicos efetuados, assim como os Atos de Enfermagem são acrescentados à Ficha Clínica de cada Utente.
* Um Funcionário, pode fazer encomendas de produtos e medicamentos.
* Os atos de levantar uma prescrição e requisitar material decrementam unidades no stock de um medicamento. Por sua vez, as encomendas de medicamentos incrementam unidades no stock.

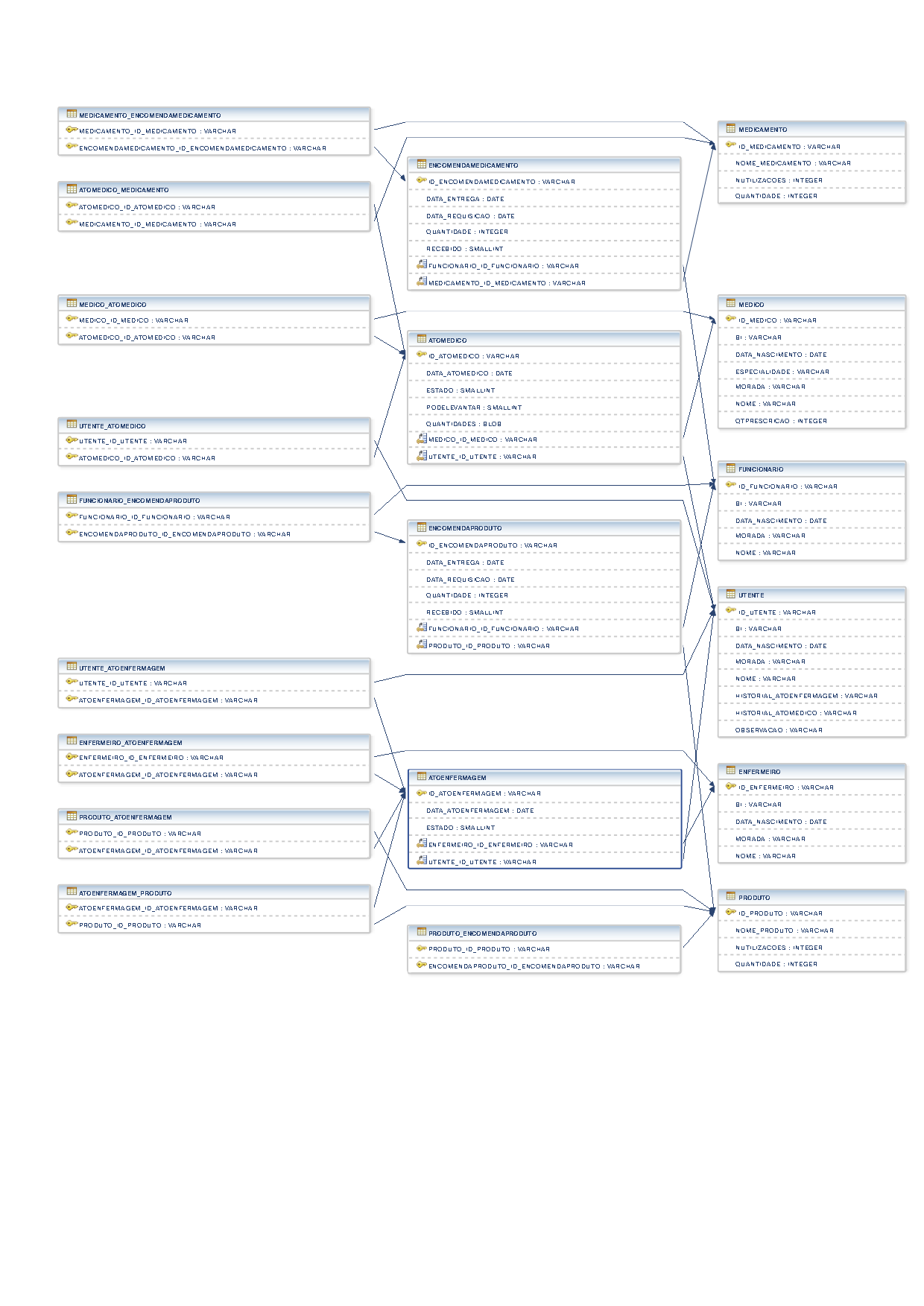


Figura 3 – Esquema da base de dados que relaciona as entidades.

Conclusão

Após concluído o este trabalho prático pode-se concluir que este tratou-se de um desafio bastante completo, na medida em que não só conjugou-se conhecimentos de JAVA como foi possível recordar conhecimentos sobre SQL.   
Como é de esperar algumas dificuldades foram aparecendo durante a elaboração deste trabalho, no entanto a grande maioria foi superada. Talvez a maior dificuldade encontrada terá sido trabalhar com a linguagem de SQL, daí que neste trabalho, muitos dos SELECTs sejam mais simples, e posteriormente seja selecionada a informação com *if’s,* e ciclos *for.*  Outra dificuldade encontrada, foi no povoamento das tabelas através do ClienteEspecial, pois quando se colocava a Main deste projeto a correr para o volume de dados referidos no enunciado, a transação era abortada ainda por volta dos 32mil utentes. No entanto quando se reduziu o volume de dados nenhum erro foi constatado.