SISTEMAS DISTRIBUÍDOS
MIEBIOM

Relatório Trabalho Prático 2

Fernando Marins 55561
Luciana Cardoso 55524
Ricardo Magalhães 52709

Resumo

O presente trabalho prático têm como objectivo a familiarização com o Java EE (Enterprise Edition) e com o servidor aplicacional glassfish. Utilizando uma arquitectura EJB (Enterprise JavaBeans) que incorpora a lógica computacional da aplicação do lado do servidor, e JPAs de forma a gerir os dados de forma a implementar no Serviço Nacional de Saúde um sistema electrónico de receitas médicas. Esse sistema permite ao Médico passar as receitas electrónicas, que estarão automaticamente disponíveis em todas as farmácias, necessitando apenas que o utente diga qual o seu número de benificiário para as mesmas poderem ser aviadas. É de salientar que cada farmácia só terá acesso às receitas que se encontem dentro do prazo da receita, impedido assim que sejam aviadas receitas fora do prazo.

Existe também um gestor que tem acesso a todos os dados e é capaz de criar novos Médicos, Utentes, Farmácias, marcação de consultas e ainda permite aceder ao estudo estatístico.

Introdução

Java EE

A tecnologia de bases de dados relacionais já existe à décadas, assim sendo hoje em dia os sistemas de bases de dados já são confiàveis.
No entanto o uso de bases de dados relacionais traz alguns inconvenientes aos programadores de linguagens orientadas a objectos como o Java. Estas duas técnologias são bastante diferentes, e o seu uso em conjunto implica enfatizar uma tecnologia em sacrificio da outra. Assim sendo o grande desafio é unir dois mundos completamente diferentes, utilizando a tecnologia relacional para armazenar objectos.

O armazenamento de objectos de um aplicação é denominado persistência de objectos, estas técnica permite que as instâncias existentes no sitema sejam armazenadas e posteriormente recuperadas, conservando-se o seu estado mesmo após a aplicação ter sido finalizada.
Para usar os recursos das bases de dados relacionais em Java é necessário fazer o que se conhece como mapeamento objecto-relacional. Neste, as classes e os atributos do sistema são

mapeados para tabelas e campos/colunas, e a persistência é feita de forma transparente pela aplicação. Assim, objectos sem memória são armazenados no banco, e objectos do banco são trazidos para a memória sempre que necessário.

O Java EE (Enterprise Edition) é uma plataforma bastante utilizada que contém um conjunto de tecnologias coordenadas que reduzem significativamente o custo e a 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 do Java EE são:

- O Enterprise JavaBeans (EJB): uma arquitetura gerida do lado do servidor utilizada para encapsular a lógica corporativa de uma aplicação. A tecnologia EJB permite o desenvolvimento rápido e simplificado de aplicações distribuídas, transacionais, seguras e portáteis baseadas na tecnologia Java.
- O Java Persistence API (JPA): uma estrutura que permite aos programadores gerir os dados utilizando o mapeamento objeto-relacional (ORM) em aplicações construídos na plataforma Java.

A arquitectura EJB surgiu de forma a reduzir o trabalho do programador, uma vez que permite que este construa menos classes e menos código, isto pois grande parte do trabalho é executada pelo sistema. Assim sendo existem várias vantagens em urtilizar este sistema. Por exemplo, são obrigatórias menos classes, já não são necessárias interfaces iniciais e de objectos, é possível a utilização de anotações para declarar componentes EJB, e quem gere as transações é o sistema. Estas anotações podem ser usadas directamente na classe para informar o sistema sobre dependências e configurações, caso não haja anotações especificas o sistema adopta a regras standard. O EBJContext permite ao programador pesquisar objectos directamente na classe. O mapeamento relacional de objectos é simplificado e transparente.

Conceitos básicos

Na JPA os objectos persistentes são denominados entidades (entities). Uma entidade é um objecto simples, que representa um conjunto de dados persistentes. Como as entidades são definidas por classes Java comuns, sem relação com frameworks ou bibliotecas, elas podem ser abstratas ou herdar de outras classes sem restrições.

Um conceito importante é que as entidades possuem um identificador (descrito pela chave primária) e estado, sendo o seu tempo de vida independente do tempo de vida da aplicação. Assim, aplicações distintas podem compartilhar a mesma entidade, que é referênciada através do seu identificador.

Para que uma entidade se torne persistente é necessário associa-la a um persistence context (contexto de persistência), que fornece a conexão entre as instâncias e a base de dados. A manipulação das entidades é feita, a partir desse contexto, por meio do entityManager (gerente de entidades), que é responsável por executar as operações básicas sobre a entidade (criação, actualização, exclusão, consultas etc.). O entity manager na JPA é uma instância da interface **javax.persistence.EntityManager.**

A implementação da JPA é feita por um persistence provider (provedor de persistência). O provedor define “como as coisas funcionam”, através da implementação de todas as interfaces definidas pela especificação da JPA. Dessa forma, cada provedor decide a maneira e o momento de carregar, actualizar e armazenar as entidades, assim como sincronizar os dados dom a base de dados. As configurações utilizadas pelo provedor numa

determinada aplicação são descritas numa persistende unit, que é configurada num arquivo especial denominado persistence.xml.

Mapeamento

As classes e interfaces da JPA estão localizadas no pacote **javax.persistence.** A API faz uso intensivo de anotações ; por isso não é necessário criar descritores XML para cada uma das entidades da aplicação. Uma entidade é uma classe Java comum, rotulada através da anotação **@Entity.** Não é preciso implementar interfaces ou estender outras classes para tornar uma classe "persistente"; a única exigência é que a classe da entidade possua um construtor sem parâmetros, pois a instanciação da classe é feita por reflexão. No código a seguir a classe Pessoa representa uma entidade. O atributo id é o identificador da entidade (chave primária), especificado através da anotação **@Id**:

```
@Entity
public class Pessoa implements Serializable {


    @Id
    @GeneratedValue
    private int id;
    private String nome;
    private String morada;
    @ManyToOne
    private CP cp;
```

Grande parte da produtividade trazida pela JPA deve-se à utilização de valores default de mapeamento, que facilitam bastante o trabalho do programador. Assim, o que não é definido explicitamente assume a configuração padrão da API. Por exemplo, por padrão a JPA considera o nome da entidade o mesmo nome da tabela no banco de dados e o nome da propriedade o mesmo nome da coluna. No código anterior, a entidade Pessoa será salva na tabela **PESSOA** e a propriedade **id** na coluna **ID**. Caso seja necessário alterar a forma de mapeamento, devemse utilizar as anotações **@Table** e **@Column**.

Os dados de uma única entidade podem estar distribuídos em mais de uma tabela, e diversos tipos de relacionamentos entre entidades são possíveis. Os mais comuns são os de agregação (anotações **@OneToOne, @OneToMany, @ManyToOne, @ManyToMany,** etc.).

Consultas

A JPA oferece suporte a consultas estáticas e dinâmicas. A API fornece uma linguagem própria de consulta, que é uma extensão bastante poderosa da EJB QL(a linguagem de consultas do EJB). Essa linguagem pode ser usada como uma alternativa ao SQL, que também é suportado. As consultas suportam polimorfismo, o que significa que quando uma entidade é consultada, todas as entidades descendentes que atendam ao critério da consulta também são retornadas. A criação de consultas é feita através do **EntityManager,** que fornece métodos específicos para instanciar consultas estáticas e dinâmicas.

As consultas estáticas possuem nomes e são descritas pela anotação NativeQuery. Elas são definidas nas entidades correspondentes e ficam "pré-compiladas". O

EntityManager utiliza o nome da consulta para instanciá-la, o que é feito através do método **createNativeQuery()**. Depois que a consulta é criada, basta introduzir os parâmetros e executá-la. A execução pode ser feita pelos métodos **getSingleResult()** ou **getResultList()**, a depender do resultado esperado. Por exemplo, basta executar a consulta conforme o exemplo abaixo:

```
javax.persistence.Query q=em.createNativeQuery("select IDRECEITA from Receita where idreceita=?1");
q.setParameter(1,new Integer(idreceita));
java.util.List<Object> l=q.getResultList();
```

As consultas dinâmicas não possuem nome, e podem ser construídas em tempo de execução. A criação desse tipo de consulta é feito através do método **createNativeQuery()**.

Ao longo de toda a realização da aplicação, sempre que se obtem uma chave primária de uma certa tabela através de um select, acede-se a qualquer informação da mesma simplesmente usando funções do java tal como:

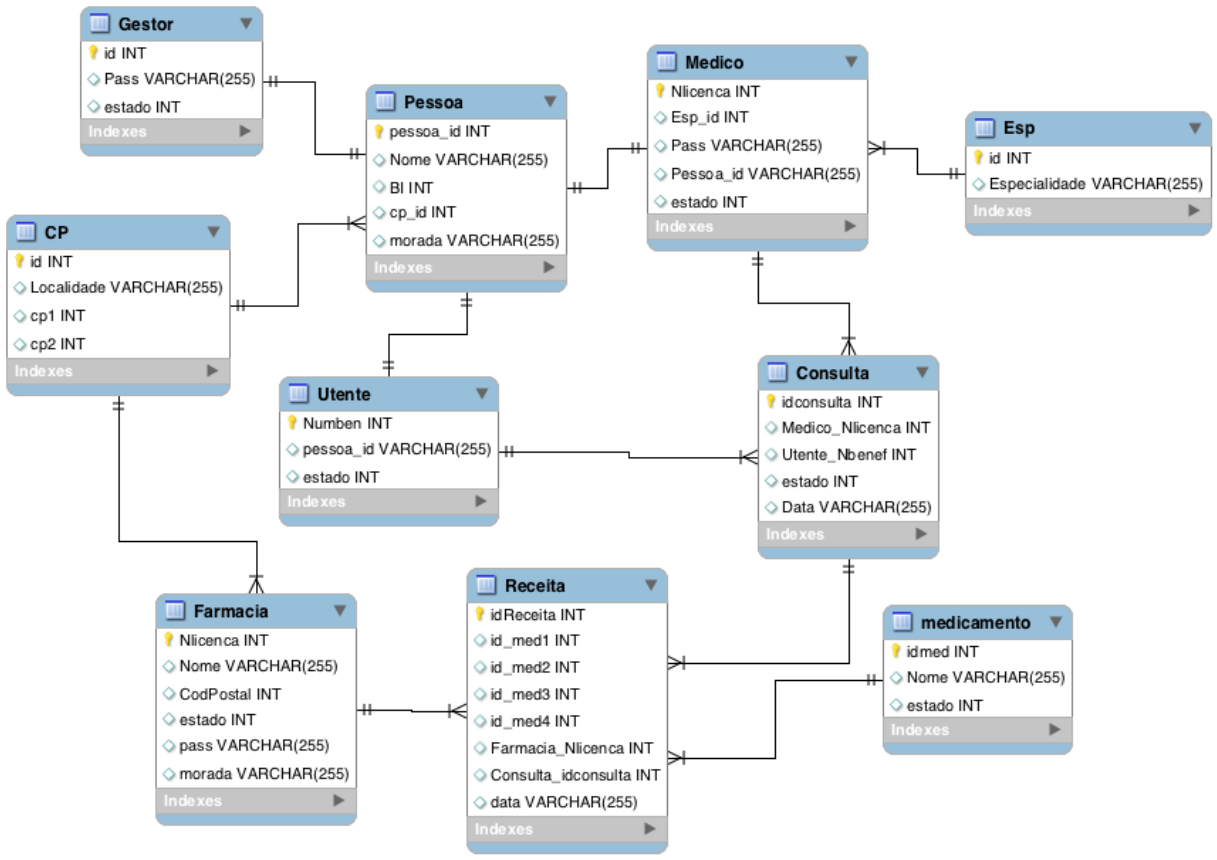
```
public int aviar_receita (int idfarmacia, int idreceita) {
    // TODO Auto-generated method stub

    Farmacia f=new Farmacia();
    f=em.find(Farmacia.class, idfarmacia);

    javax.persistence.Query q=em.createNativeQuery("select IDRECEITA from Receita where idreceita=?1");
    q.setParameter(1,new Integer(idreceita));
    java.util.List<Object> l=q.getResultList();
    if(l.size()==0){
        return -1;
    }

    Receita r=new Receita();
    r=em.find(Receita.class, idreceita);

    r.setFarmacia(f);
    f.addReceita(r);

    return 1;
    }
```

Arquitectura

Relativamente à arquitectura foi decidido construir 10 tabelas de dados, ligadas entre si por relações de um para um, um para muitos, muitos para um ou muitos para muitos. Tal como referido anteriormente, cada tabela necessita de ser identificada como **@Entity**.

No total utilizaram-se 5 session Beans, uma para logins, uma para os gestores, uma para os médicos, uma para as farmácias e uma para as estatísticas, nestas sessions beans encontram-se todas as funções realizadas e estas serão invocadas por uma “main” que contém um menu de navegação por toda a aplicação. É necessário referir que nem todos os utilizadores têm acesso à totalidade da aplicação.

Só o gestor pode criar médicos, utentes e farmácias, este tem ainda a função de marcar consultas e pode aceder às estatísticas. O médico pode também marcar consultas, dar consultas e passar receitas. A farmácia tem unicamente a função de aviar receitas.



Implementação

Para implementar a aquitectura supra referida teve-se que tomar várias decisões. A primeira grande decisão foi criar uma tabela “PESSOA” que possui todos os intervenientes do sistema quer sejam utentes, médicos, gestores, ou utentes e médicos em simultâneo, gestores e utentes...

Focando na parte de inserção de um novo médico na base de dados, quando no menu do gestor escolhemos essa opção, se a pessoa já existir no sistema é apenas necessário inserir a especialidade e o número de licença de médico. Caso a pessoa não exista na tabela “PESSOA”, o código encaminha para a situação em que “cria uma nova pessoa”, pedindo ao utilizador todos os argumento necessários a inserção da mesma. Além de reconhecer uma pessoa que já existe na base de dados, o código também identifica as especialidades dos médicos já existente e códigos postais, e se não existirem também são criados. Todas estas características torna este sistema automático e persistente (o utilizador só acrescenta informação só quando necessário).

O cliente ao correr a classe “Main” acede a um menu principal onde escolhe o tipo de utilizador e faz o login, toda este classe é preenchida de funções que garantem a automaticidade anteriormente referida. Após a invocação dessas funções presentes na classe “Main”, estas invocam outras funções do lado do servidor (nas sessions), que farão operações sobre a base de dados e devolverão os resultados pretendidos. Cada session

necessita ter uma interface remote que declara as funções que podem ser acedidas dentro da session.

Em relação às prioridades é de se salientar que somente uma pessoa com o estatuto de gestor pode fazer operações bastante poderosas como inserir novos códigos postais, especialidades, médicos, etc.

Estatísticas

De forma a saber quem emite as receitas, quais os medicamentos prescritos, quem é o beneficiário e também qual a farmácia responsável pelo fornecimento de medicamentos, realizou-se um conjunto de estatísticas acessíveis unicamente por utilizadores com permissão de gestor, relativas a médicos, farmácias, utentes e medicamentos.

Alguns exemplos destas são:

- Quantas receitas emitiu no mês de Janeiro de 2012 o médico com o número de licença 1:

```
Acerca dos Medicos
Escolha o numero que corresponde a questao que deseja:
(1) Quantas receitas emitem por dia, mes e ano?
(2) Quais os medicamentos que cada medico mais prescrebe?
(3) Quais as farmacias que despacham as receitas?
1
(1) Quantas receitas por dia?
(2) Quantas receitas por mes?
(3) Quantas receitas por ano?
2
Mes (mm):
1
Ano (yyyy):
2012
Número de licença do médico:
1
Médico: 1
Mês: 2012-1
Quantidade de receitas nesse mês: 5
Deseja fazer mais estudos estatísticos? (S/N)
```

- Quais os medicamentos que o médico com licença número 1 mais prescrebe?

```
Acerca dos Medicos
Escolha o numero que corresponde a questao que deseja:
(1) Quantas receitas emitem por dia, mes e ano?
(2) Quais os medicamentos que cada medico mais prescrebe?
(3) Quais as farmacias que despacham as receitas?
2
Número de licença do médico:
1
Médico: 1
Medicamentos mais vendidos:
Quantidade       Medicamento
5                 aspirina
3                 benuron
2                 yasmin
2                 nasex
1                 zovirax
1                 libeli
1                 brufene
1                 aspegic
0                 voltaren
0                 trifene
0                 lisaspin
0                 ilvico
Deseja fazer mais estudos estatísticos? (S/N)
```

- Quais são as farmácias que despacham mais receitas?

```
Acerca das Farmacias
Escolha o numero que corresponde a questao que deseja:
(1) Quais as farmacias que despacham mais receitas?
(2) Quais os medicamentos mais vendidos por cada farmacia?
1

Farmácia         Quantidade de receitas aviadas
farmlima        4
Farmacia da fonte       4
Central 2
Saudiva 1
Espetaculo      1
Deseja fazer mais estudos estatísticos? (S/N)
```

- Quais são os medicamentos mais vendidos?

```
Acerca dos Medicamentos
Escolha o numero que corresponde a questao que deseja:
(1) Quais os medicamentos mais vendidos?
(2) Quais as farmacias que mais vendem um determinado medicamento?
1

Quantidade vendida       Medicamento
8       aspirina
5       voltaren
4       yasmin
4       nasex
3       benuron
2       zovirax
2       trifene
2       libeli
2       ilvico
1       lisaspin
1       brufene
1       aspegic
Deseja fazer mais estudos estatísticos? (S/N)
```

- Quais são as farmácias que vendem mais o medicamento com id 10?

```
Acerca dos Medicamentos
Escolha o numero que corresponde a questao que deseja:
(1) Quais os medicamentos mais vendidos?
(2) Quais as farmacias que mais vendem um determinado medicamento?
2
Id medicamento:
10
Quantidade       Farmácia

5       Farmacia da fonte
2       Central
1       farmlima
0       Saudiva
0       Espetaculo
0       Costa e Lima
Deseja fazer mais estudos estatísticos? (S/N)
```