DOI: http://dx.doi.org/10.4025/revtecnol.v24i1.19232

UMA EXPERIÊNCIA DE ADOÇÃO DO *JAVA 6 WEB PROFILE* NO DESENVOLVIMENTO DE UM SISTEMA PARA GERENCIAMENTO DE UMA CLÍNICA DE PSICOLOGIA

AN ADOPTION EXPERIENCE USING JAVA 6 WEB PROFILE TO DEVELOP A PSYCHOLOGY CLINICAL MANAGEMENT SYSTEM

Ana Paula Allian[1]
Edson A. Oliveira Junior[2]

Resumo: Os sistemas de informações gerenciais (SIGs) tornaram-se um dos principais componentes para gerenciar os processos de negócios das empresas. Um registro organizado e detalhado dos processos rotineiros de uma clínica como, por exemplo, controle de pacientes, agenda e controle financeiro, fornecem, instantaneamente, informações importantes sobre o desenvolvimento da empresa, incluindo dados financeiros e dados sobre pacientes. Este artigo apresenta o desenvolvimento de um sistema de gerenciamento para clínicas de psicologia, utilizando a especificação Java EE 6 (Java Enterprise Edition 6) conhecida por Web Profile. Esse perfil consiste em uma seleção de tecnologias para a construção de aplicações Web lightweight comparada à especificação Java EE 6 completa. O artigo apresenta o Web Profile, utilizando uma abordagem prática, na qual uma aplicação Web completa é desenvolvida, empregando tecnologias presentes no Web Profile como, por exemplo, EJB Lite, JPA, CDI e JSF. O artigo é relevante para desenvolvedores que trabalham ou desejam trabalhar com a implementação de aplicações Web utilizando padrões definidos na especificação Java EE 6.

Palavras-chave: Clínica de psicologia. Java EE 6. Web Profile. EBJ Lite. Sistema de gerenciamento.

Abstract: Management information systems have become one of the main issues for managing the organizations business processes. An organized and detailed record of the routine processes of a clinic as, for instance, patient records, schedule, financial management, provides essential information on a company development, which includes both financial and patient's data. This paper presents the development of a system for managing psychology clinics by using Java EE 6 specification (Java Enterprise Edition 6 Web Profile), which is a subset of technologies for building Web applications more lightweight than the complete Java EE 6 profile specification. This paper presents Web Profile using a practical approach, in which a simple and complete Web application is developed using Web Profile technologies, such as, EJB Lite, JPA, CDI, and JSF. This paper is useful for developers who are intended to work with application development adopting Web Profile defined in the Java EE 6 specification.

Keywords: Psychology Clinic. Java EE 6. Web Profile. EBJ Lite. Software Management.

1 INTRODUÇÃO

O desenvolvimento de aplicações Web é um assunto que continua em evidência no mercado de software. Entre outros fatores, isto se deve ao crescente número de usuários acessando a Internet, o que faz com que as empresas cada vez mais procurem divulgação por meio de sites para possíveis clientes (Molinari, 2012).

Com o advento do Java EE 6 (*Java Enterprise Edition*) muita coisa mudou na Web e os *frameworks* ficaram mais versáteis. Logo, se fez necessária uma nova versão do Java EE 6,

[1] Aluna do curso de especialização em Desenvolvimento de Sistemas para Web – Universidade Estadual de Maringá (UEM) - Av. Colombo, 5790 – Bloco C56 – Maringá – PR – Brasil – ana.allian@gmail.com
[2] Departamento de Informática – Universidade Estadual de Maringá (UEM) - Av. Colombo, 5790 – Bloco C56 – Maringá – PR – Brasil - edson@din.uem.br

totalmente reescrita, voltada especificamente para facilitar o aprendizado e o desenvolvimento de aplicações corporativas (Sampaio, 2011). O grande desafio do Java EE 6 foi tornar a sua plataforma mais flexível, ao mesmo tempo em que foram adicionadas novas especificações. Assim, o Java EE 6 está concentrado em fornecer maior simplicidade à plataforma, por meio da introdução de perfis e da adoção do processo de exclusão (também chamado de marcação para exclusão) de tecnologias obsoletas e mais complexas. Tal processo consiste na proposição de uma lista de funcionalidades para possível remoção no Java EE 7 como, por exemplo, EJB 2.x, JAX-RPC, JAXR (Gonçalves, 2011).

É com base nos perfis do Java EE 6 e, utilizando a IDE Eclipse (Eclipse, 2012), o servidor de aplicação GlassFish 3.1.1 (GlassFish, 2012) e o sistema gerenciador de banco de dados MySQL (MySQL, 2012), que este artigo apresenta o *Web Profile* de forma prática, na qual uma aplicação Web com poucas características, porém completa (desde a especificação até a implementação) é desenvolvida, empregando tecnologias presentes no *Web Profile*, como EJB *Lite*, JPA, CDI e JSF (Souza, 2012).

Este artigo está organizado da seguinte forma: na seção 2 são apresentados os principais conceitos sobre *Web Profile*, *EJB Lite* e outras tecnologias; na seção 3 é apresentada a implementação do sistema desenvolvido para gerenciar clínica de psicologia e a sua arquitetura; a seção 4 apresenta as lições aprendidas e a seção 5 apresenta as conclusões e direções para trabalhos futuros.

2 TECNOLOGIAS ADOTADAS

2.1 WEB PROFILE

A plataforma Java EE contém um conjunto de tecnologias coordenadas que reduz significativamente o custo e a complexidade do desenvolvimento, implantação e gerenciamento de aplicativos de várias camadas centradas no servidor. À medida que a plataforma foi evoluindo, o número de tecnologias agregadas cresceu para fornecer diferentes serviços e atender a diferentes tipos de aplicações corporativas. Tal evolução, no entanto, causou um problema: para muitas aplicações os conjuntos de tecnologias oferecidos são complexos demais, consumindo recursos desnecessários do servidor, tais como, maior alocação de memória e maior consumo de processamento do CPU (Souza, 2010).

Para tornar a plataforma menos complexa, o grupo do Java EE 6, formado pelo *Java Community Process* (JCP), responsável por descrever as especificações propostas e as tecnologias a serem adicionadas à plataforma Java (Java *Community*, 2012), introduziu o *Web Profile, este consiste de* um subconjunto de *Application Programming Interface* (APIs) do Java EE 6, visando o seu uso no desenvolvimento de aplicações corporativas (Gonçalves, 2011). O *Web Profile* inclui suporte a tecnologias na camada Web, como *Java Server Faces* (JSF) (Geary e Hortsmann, 2010), *Facelets* (Geary e Hortsmann, 2010), *Java Server Pages* (JSP) (Hall e Brown, 2000), e *Servlet* 3.0 (Heffelfinger, 2010). Também traz funcionalidades de validação de classes de dados (*beans*), *Java Persistence API* (JPA 2) (Heffelfinger, 2010) para persistência, *Java Transaction API* (JTA) (Panda e Rahman, 2007) para o gerenciamento de transações, EJB 3.1 *Lite* (Ort, 2012) para serviços na camada de negócios e CDI(Ort, 2012), que descreve um modelo de componentes geral baseado na injeção de dependências. Porém, o *Web Profile* só inclui suporte para arquivos Web (.WAR) e não para arquivos *Enterprise* (.EAR) (Souza, 2010).

A Figura 1 apresenta todas as APIs que compõem o Java EE 6 e, em destaque o subconjunto de tecnologias que formam o Web *Profile*, a base de estudo deste artigo.

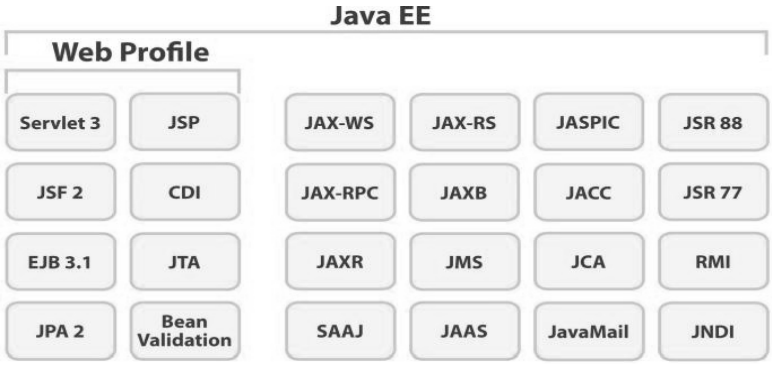


Figura 1. APIs Java EE 6 (Rahman, 2012)

2.2 EJB LITE

Os *Enterprise Java Beans (*EJBs) são componentes de servidor que encapsulam lógica funcional e cuidam de transações e segurança. Os EJBs usam um modelo de programação que combina facilidade de uso e robustez (Gonçalves, 2011). O objetivo do EJB é simplificar o desenvolvimento fazendo uso de anotações de metadados para gerar vários artefatos, reduzindo a complexidade e ao mesmo tempo em que permite a escalabilidade para aplicações empresariais. Tudo isso ocorre como resultado da anotação de um único *Plain Old Java Object* (POJO), que é distribuído em um *Container* (Gonçalves, 2011). Um *container* de EJB é um ambiente de tempo de execução que oferece serviços, tais como, gerenciamento de transações, controle de concorrência, agregação e autorização de segurança (Gonçalves, 2011).

Os EJBs representam o modelo de componente predominante no Java EE 6 (Rubinger e Burke, 2010). Por essas razões, a especificação JAVA EE 6 definiu um subconjunto mínimo de API de EJB completa, conhecida como EJB *Lite*. Tal API inclui uma pequena e poderosa seleção de funcionalidades de EJB, convenientes para escrita de lógica funcional portável, segura e transacional. Qualquer aplicação EJB *Lite* pode ser distribuída em qualquer *Container* Java que implemente a API EJB 3.1. O EJB *Lite* é composto do subconjunto da API de EJB listada na Tabela 1 (Gonçalves, 2011).

Tabela 1. Comparação entre a EJB Lite e a EJB Completa (Gonçalves, 2011)

Características	*EJB Lite*	*EJB*
Beans de Sessão sem Estado	Sim	Sim
Beans de Sessão com Estado	Sim	Sim
Beans de Sessão Singular	Sim	Sim
Beans de Sessão Dirigidos por Mensagens	Sim	Sim
Beans de entidades 1.x/2.x		Sim
Vista sem interface	Sim	Sim
Interface Local	Sim	Sim
Interface Remota		Sim
Interface 2.x		Sim
Serviços Web JAX-WS		Sim
Serviços Web JAX-RS		Sim
Serviços Web JAX-RPC		Sim
Serviço temporizador		Sim
Chamadas Assíncronas		Sim
Interceptadores	Sim	Sim
Interoperabilidade RMI/IIOP		Sim
Suporte a transação	Sim	Sim
Segurança	Sim	Sim
API Embutível	Sim	Sim

Com base na Tabela 1, nota-se que o EJB *Lite* inclui *beans* de sessão sem estado (*Stateless*), *beans* de sessão com estado (*Stateful*), *beans* de sessão que guardam estado em nível de aplicação (*Singleton)*, transações declarativas e programáticas, além de segurança declarativa e programática (Rahman, 2012).

Beans de sessão encapsulam lógica funcional, são transacionais, se baseiam em um *Container* que faz agregação, multissegmentação e segurança (Gonçalves, 2011). Para implementar tal componente é necessária apenas uma classe Java e uma anotação, como representado na Quadro 1:

```
@Singleton
public class ConvenioDao {

@PersistenceContext
private EntityManager entityManager;

   public void inserir(Convenio c) {
      entityManager.persist(c);
      entityManager.flush();
   }
}
```

Quadro 1. Classe ConvenioDao

Como mostra o código da Listagem 1, a classe não implementa nenhuma interface, mas sim, uma classe Java num componente transacional e seguro: @Singleton. Usando-se o gerenciador de entidades (*EntityManager*), a classe insere uma instância de Convenio na base de dados de uma maneira simples.

Simplicidade também é aplicada ao lado cliente. A invocação de um método na classe ConvenioDao só exige uma anotação para obtenção de uma referência, usando-se injeção de dependência (Gonçalves, 2011). A injeção de dependência permite que um *Container* injete automaticamente uma referência à classe ConvenioNegocio pela anotação do atributo @EJB, como mostra a Quadro 2.

```
@Singleton
public class ConvenioNegocio {

        @EJB
        private ConvenioDao repositorio;

        public void inserir(Convenio c) {
                if(c.getNome().trim().equals("")||c.getNome()==null){
                        throw new Exception("Digite o nome");
                }
                repositorio.inserir(c);
        }
}
```

Quadro 2. Classe ConvenioNegocio

A classe ConvenioNegocio não cria uma instância de ConvenioDao, mas obtém primeiro uma referência ao EJB, seja por injeção ou por busca JNDI e depois chama o método.

2.3 GLASSFISH 3.1.1 - OPEN SOURCE WEB PROFILE

GlassFish é um servidor de aplicação *Open Source* e é a implementação de referência (RI) para a tecnologia Java EE (Gonçalves, 2011). O principal objetivo do GlassFish 3 é a modularização das funcionalidades centrais. Essa modularidade e extensibilidade permitem que o GlassFish 3 seja desde um simples servidor Web, na escuta de comandos administrativos, até um ambiente de execução mais capaz, pela simples distribuição de artefatos, tais como, arquivos WAR ou arquivos JAR EJB. Além disso, o servidor básico inicia em apenas alguns segundos, tornando-se um ambiente amigável para o desenvolvedor (Heffelfinger, 2010).

O console de administração (*Admin Console*) é uma interface de usuário para administração baseada em navegadores (Figura 2) para o servidor de aplicações. Esta ferramenta serve para administradores e desenvolvedores. O Console apresenta representações gráficas dos objetos sob gerenciamento, visualização melhorada dos arquivos de registros, *status* do sistema e monitoração de dados (Gonçalves, 2011). O console de administração está disponível na inicialização do GlassFish, em http://localhost:4848.

A versão *Web Profile* do *GlassFish* contém tecnologias para Web que fazem parte da plataforma completa e é projetado para desenvolvedores que não precisam do conjunto completo de APIs do Java EE. Esse perfil também é instalado com o *Web Profile* do Java EE 6 SDK (Oracle, 2012).

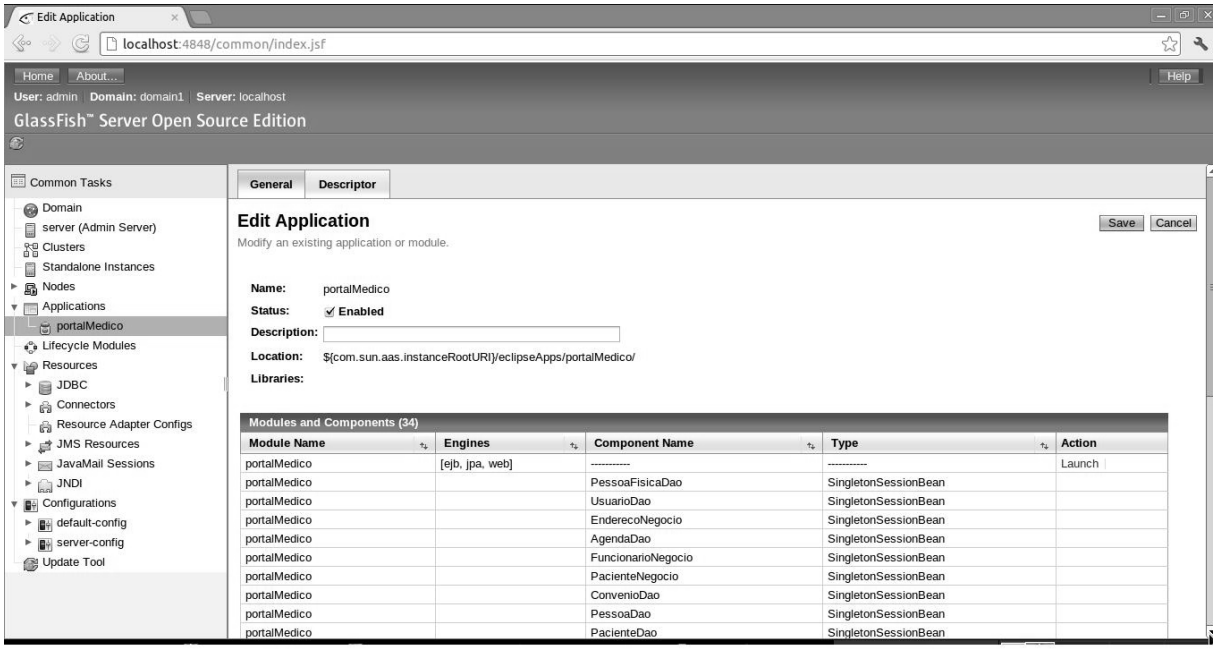


Figura 2. Console de Administração do GlassFish

2.4 MySQL

MySQL é um Sistema Gerenciador de Bancos de Dados (SGBD) relacional que utiliza a linguagem padrão *Structured Query Language* (SQL) e é amplamente utilizado em aplicações para a Internet. É um dos mais populares bancos de dados (Niederauer, 2006).

O MySQL á uma alternativa atrativa, pois possui uma tecnologia complexa de banco de dados e, além disso, é gratuito. Têm como destaque suas características de velocidade, escalabilidade e confiabilidade, o que vem fazendo com que seja adotado por departamentos de TI, desenvolvedores Web e vendedores de pacotes de softwares (Niederauer, 2006).

3 SISTEMA WEB PARA GERENCIAMENTO DE CLÍNICAS DE PSICOLOGIA

Esta seção apresenta o sistema desenvolvido para gerenciar as informações de uma clínica de psicologia. A clínica iniciou suas atividades há dois anos e o desenvolvimento de um sistema foi proposto para agilizar e organizar os processos de controle de pacientes. Nesta seção é apresentada a rotina da clínica e também o processo para o desenvolvimento da aplicação. As subseções correspondem à descrição da empresa, dos procedimentos atualmente utilizados, da modelagem do projeto e da arquitetura do sistema proposto, como uma experiência na adoção do Web Profile.

3.1 DESCRIÇÃO GERAL DA EMPRESA

O Consultório de Psicologia está situado em Londrina/PR há dois anos, sendo especializado em Psicologia Clínica e se dedica ao estudo dos transtornos mentais e dos aspectos psíquicos de doenças não mentais. Atualmente, conta com dois psicólogos que gerenciam o controle de pacientes manualmente por meio de agendas, planilhas eletrônicas e folhas impressas.

A clínica de psicologia funciona em três turnos, manhã, tarde e noite e, abrange uma faixa etária diversificada – crianças, adolescentes, jovens, adultos e idosos. A média de atendimento mensal é de 100 pacientes que chegam ao consultório espontaneamente ou encaminhados por outras instituições.

3.2 MODELAGEM DO SISTEMA PROPOSTO

A primeira etapa de desenvolvimento do sistema proposto remete à elicitação de requisitos. Assim, para documentar os principais requisitos do sistema foi utilizada a técnica de casos de uso (Booch et al, 2006).

Após entrevistar os psicólogos, foi possível desenvolver o modelo de caso de uso da Figura 3, o qual apresenta os principais requisitos funcionais do projeto. Serão omitidos os casos de uso de cadastro, alteração, atualização e exclusão de informação conhecidos por CRUD.

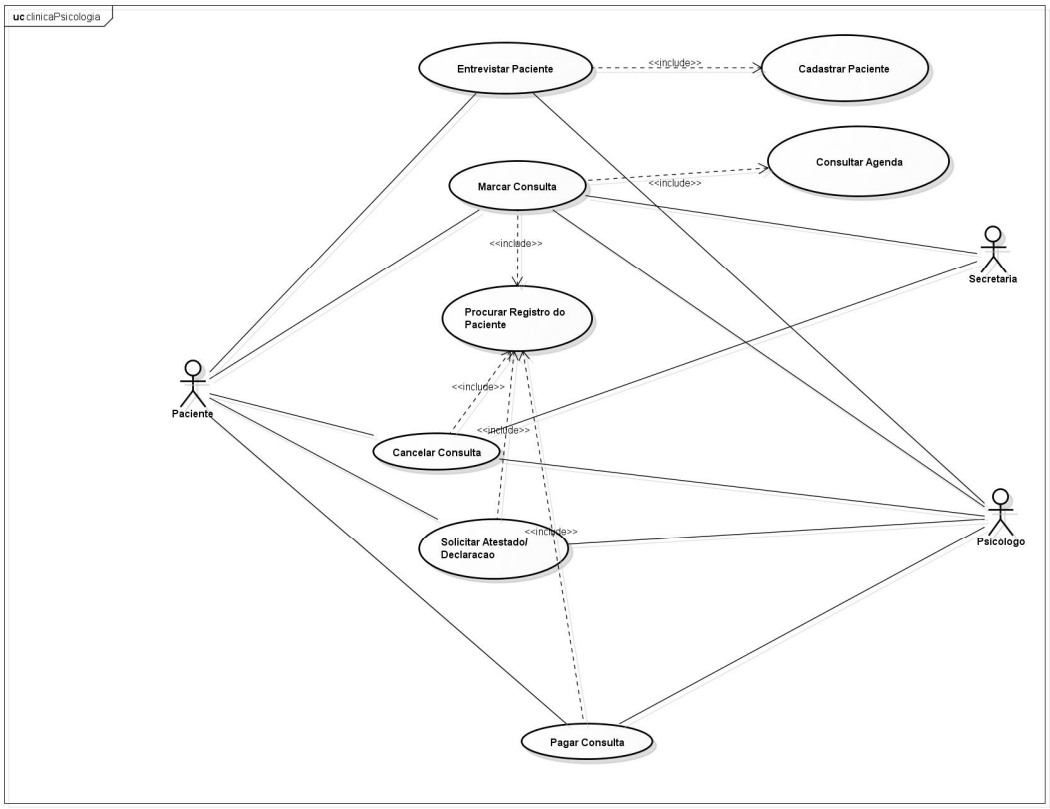


Figura 3. Diagrama de Caso de Uso

No modelo de casos de uso da Figura 3, o Psicólogo realiza o caso de uso “Entrevistar Paciente” por meio da entrevista com o Paciente, o que inclui “Cadastrar Paciente”. O psicólogo ou secretária pode consultar a agenda e definir as datas com o paciente para agendamento da consulta. Se o paciente desistir da consulta ou o psicólogo não puder atender no dia agendado, o psicólogo ou secretária cancela o agendamento. Se o paciente comparecer à consulta, o psicólogo registra as informações da consulta no sistema e solicita o pagamento pela consulta. O paciente realiza o pagamento e o psicólogo registra o valor recebido no sistema. Se necessário, paciente solicita atestado e/ou declaração. Para isso, o psicólogo acessa o sistema e emite a declaração e/ou atestado para o paciente.

Com base no modelo de casos de uso da Figura 3, foi proposto o diagrama de classes de domínio conforme ilustrado na Figura 4. Tal diagrama é um dos mais importantes diagramas da UML e, representa a arquitetura do sistema proposto, permitindo a elaboração de outros diagramas.

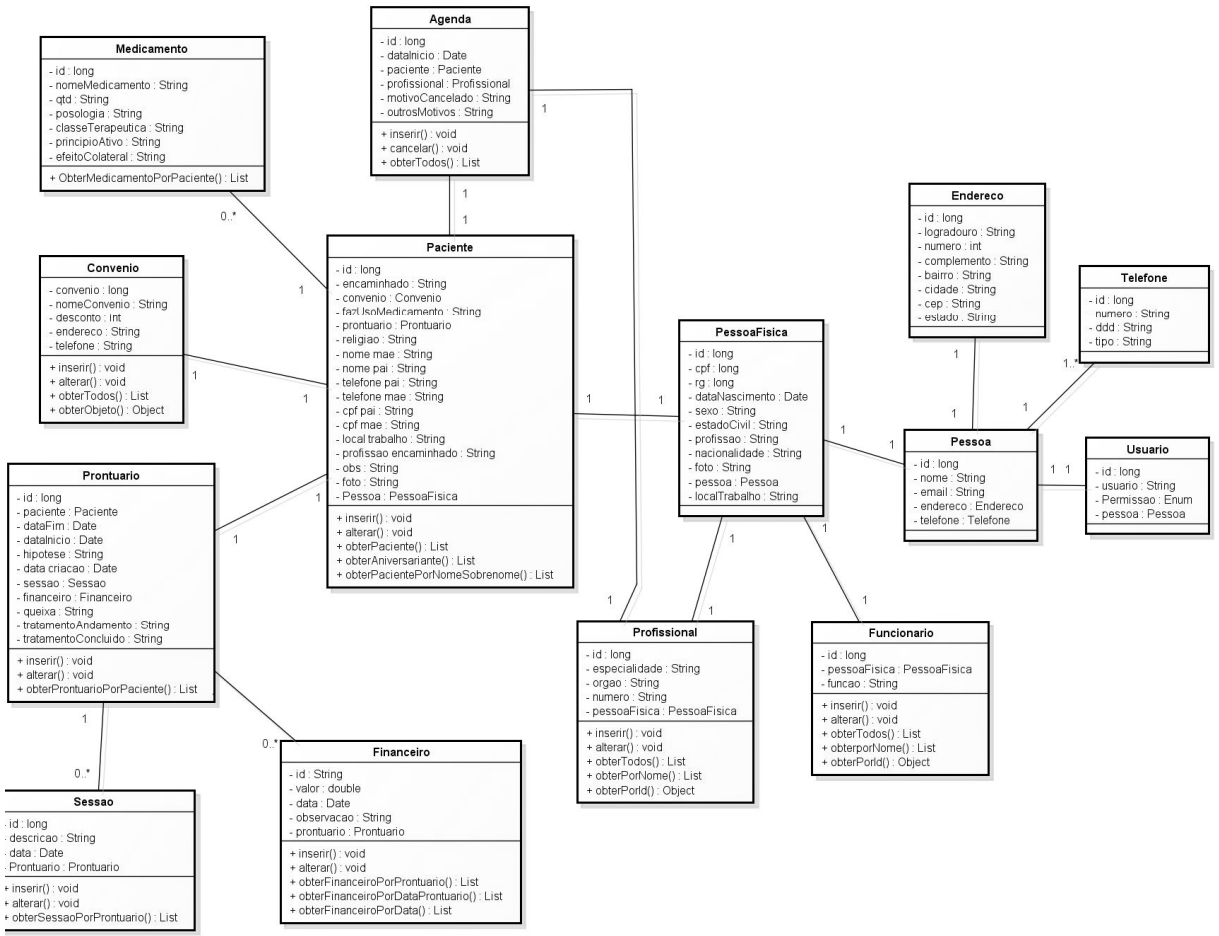


Figura 4. Diagrama de Classes de Domínio

O diagrama de classes de domínio contém todos os relacionamentos dos objetos criados para o desenvolvimento do sistema. Na análise da Figura 4, as classes estão relacionadas por meio de associações. A classe Paciente está associada à classe Prontuário em um relacionamento 1 para 1; Prontuário é uma das principais classes do projeto pois, envolve o controle diário das consultas do paciente e o cadastro financeiro. A classe Paciente também está associada à classe PessoaFisica. Essa última é outra importante classe do diagrama, com uma associação unitária. Observa-se que PessoaFisica está associada à Pessoa, o que possibilita futuramente, a criação de PessoaJuridica, sem interferir na lógica do projeto.

Pode-se observar por meio do diagrama de classes do projeto (Figura 5) o comportamento do padrão Transaction Script (Panda *et al*, 2007). Observa-se que a lógica de domínio da aplicação está separada por classes e, cada classe possui procedimentos para tratar as transações. Esse diagrama é uma representação das interações entre as classes do sistema. As interações são as chamadas dos métodos existentes em cada classe o que evidenciam a existência de um fraco acoplamento. Fraco acoplamento é a capacidade de uma classe em herdar o comportamento de outra, esta é uma das principais características do paradigma de orientação a objetos.

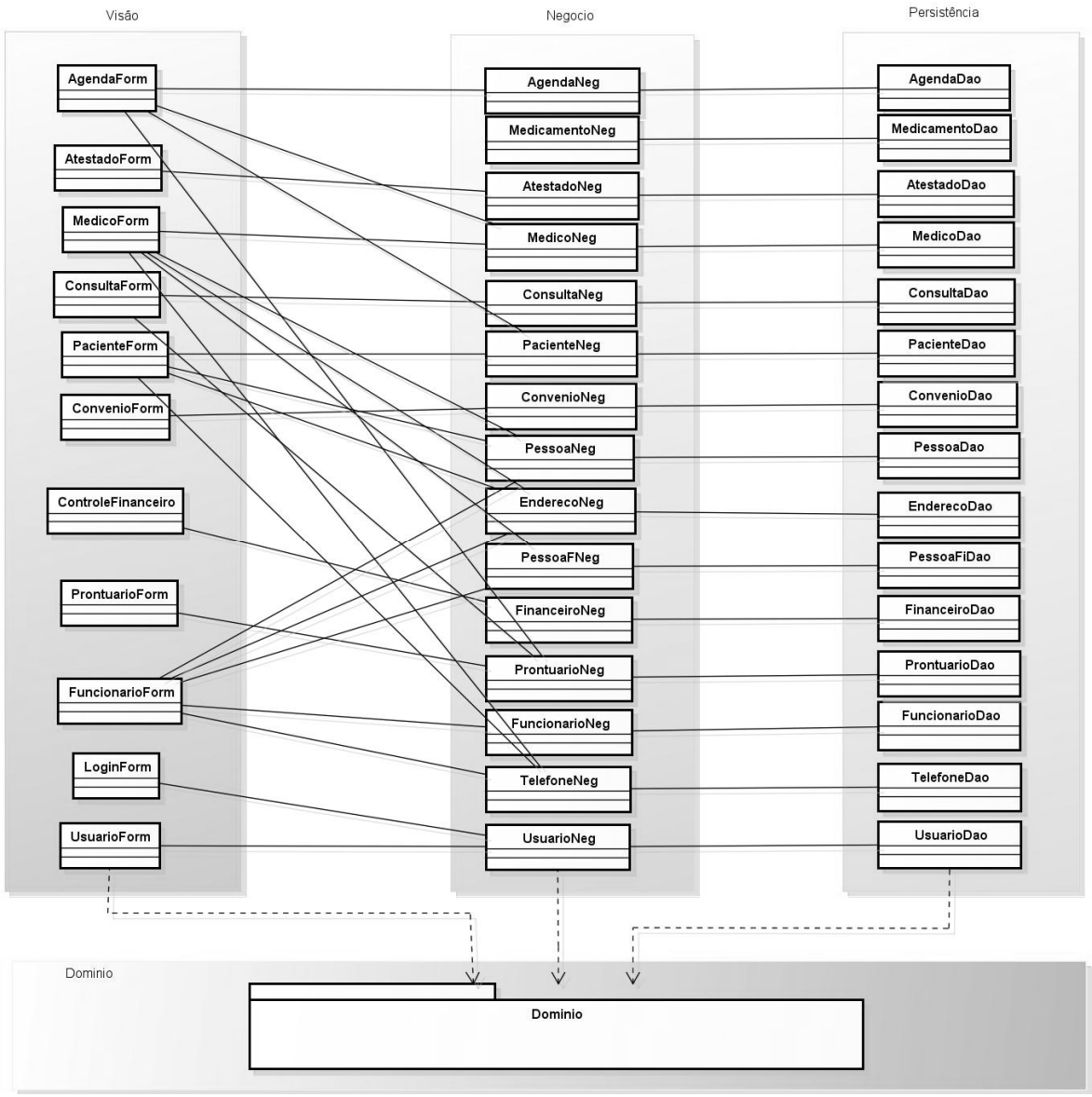


Figura 5. Diagrama de Classes do Sistema Proposto

3.3 ARQUITETURA DO SISTEMA PROPOSTO

A arquitetura do sistema proposto segue o padrão *Model-View-Controller* (MVC). Tal modelo isola a lógica de negócio do sistema da camada de visão. Na camada de domínio foi utilizada a *Java Persistence API* (JPA) (Heffelfinger, 2010) que define um meio de mapeamento para objetos Java simples e também gerencia o desenvolvimento de entidades do modelo relacional. A Figura 6 apresenta um exemplo de entidade de domínio com mapeamento em JPA.

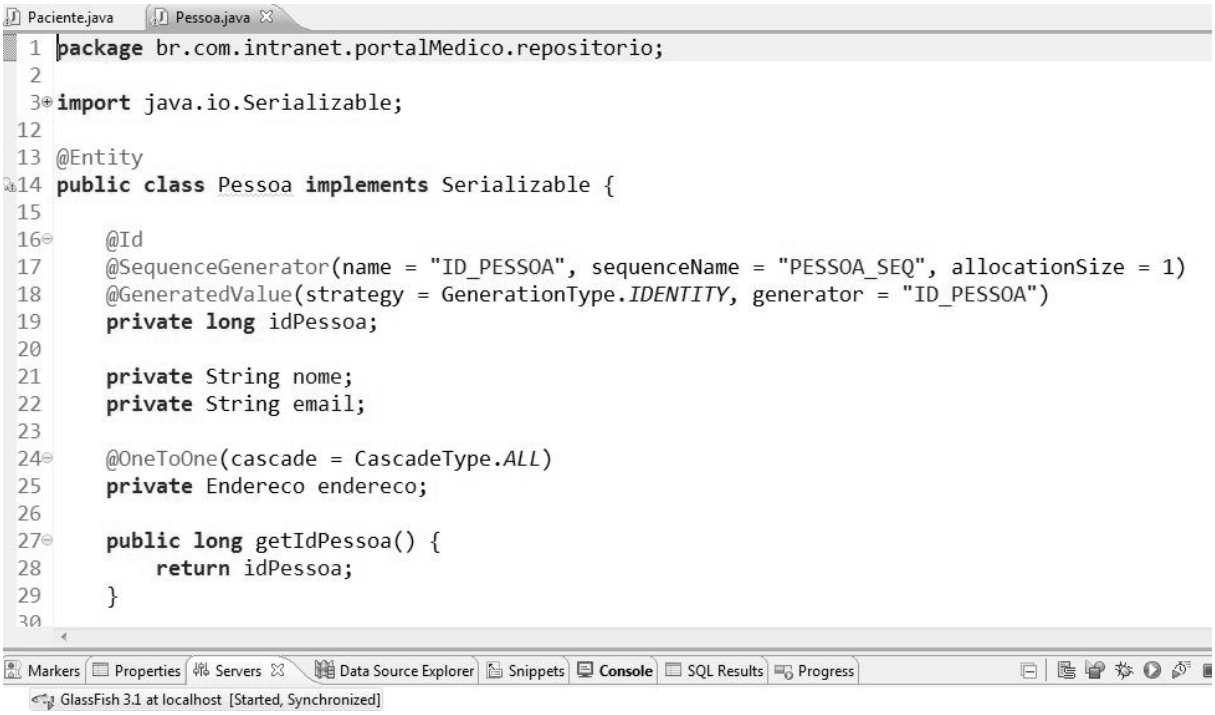


Figura 6. Mapeamento da Entidade Java Pessoa com JPA

A principal anotação na camada de domínio é *@Entity* (linha 13 da Figura 6). Essa anotação define que o objeto modelado é uma entidade em um banco de dados. *@Id* informa qual atributo é a chave primária da tabela. Observa-se também o uso de anotações de relacionamento *@OneToOne* (linha 24 da Figura 6) com a entidade Endereco.

Na camada de persistência foi utilizado o *EntityManager*, este é responsável por praticamente todas as operações de persistência de objetos e é de fácil compreensão e utilização. A anotação *PersistenceContext* (linha 17 da Figura 7) permite o uso de uma área de memória que mantém os objetos que estão sendo manipulados pelo *EntityManager*. Para gerenciar as transações foi utilizada a *Java Transaction API* (JTA) (Panda et al, 2007), a qual não cria nenhuma transação, mas utiliza as que já existem.

Figura 7. Exemplo de Persistência de Dados com EntityManager

Na camada de negócio fez-se uso do padrão *Transaction Script* (Panda et al, 2007).

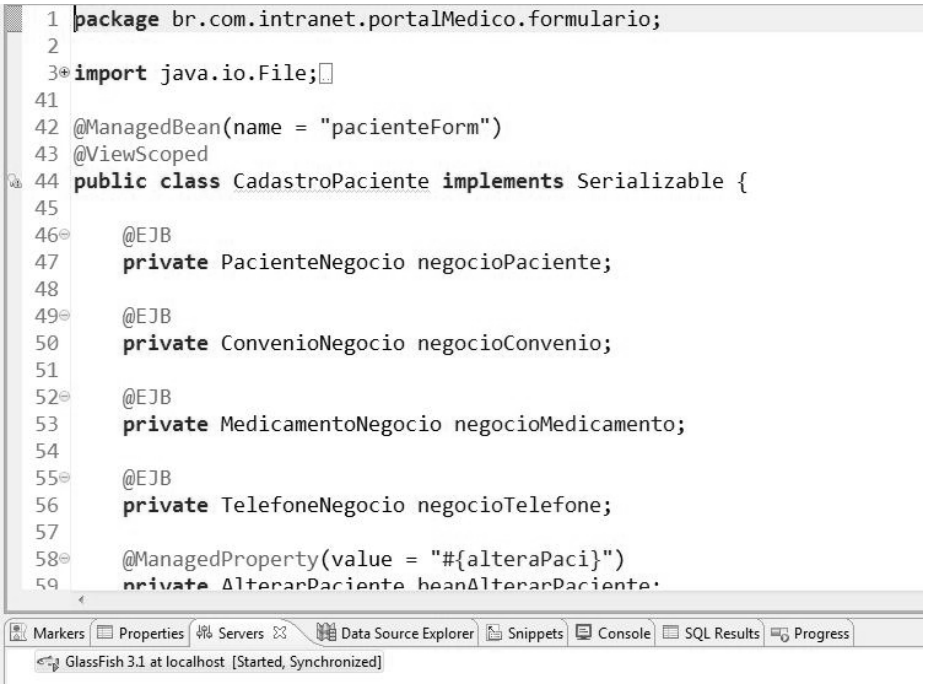
Tal padrão organiza a lógica de negócio em procedimentos. Cada procedimento trata um único pedido da camada de apresentação. O procedimento recebe requisições da camada de apresentação, as processa com validações e cálculos, armazena dados em um banco de dados e/ou invoca operações de outros sistemas. Em seguida, entrega dados novamente à camada de apresentação, efetuando eventualmente mais cálculos para organizar e formatar a resposta.

Para gerenciar os objetos criados pelo sistema foi utilizado o EJB 3.1 *Lite*, um subconjunto de APIs do EJB específicas para aplicações Web. A utilização do *Session Bean Singleton* (linha 11 da Figura 8) fornece uma garantia extra à aplicação, pois o *bean* será criado apenas uma vez pela máquina virtual Java (JVM) em execução.

```
 3 import java.util.List;
10
11 @Singleton
12 public class ConvenioNegocio {
13
14     @EJB
15     private ConvenioDao repositorio;
16
17     public void gravar(Convenio t) throws NegocioException {
18         if (t.getNomeConvenio().trim().equals("") || t.getNomeConvenio() == null) {
19             throw new NegocioException("Digite o nome do convenio");
20         }
21         try {
22             repositorio.inserir(t);
23         } catch (Exception e) {
24             throw new NegocioException(e.getMessage());
25         }
26     }
27     public List<Convenio> obterTodosConvenio() throws NegocioException {
28         try {
29             return repositorio.obterTodos();
30         } catch (Exception e) {
31             throw new NegocioException(e.getMessage());
32         }
33     }
34
```

Figura 8. Camada de Negócio usando Session Bean Singleton

Na camada de visão foi utilizado um *framework* MVC de aplicações Web baseado em Java conhecido por JSF. O JSF se destina a simplificar o desenvolvimento de interfaces de usuários baseados em Web. A Figura 9 apresenta um *Managed Bean*, responsável por intermediar a comunicação entre páginas (.xhtml) e o modelo da aplicação Java (Geary e Hortsmann, 2010). O uso do escopo *@ViewScoped* é utilizado para indicar que o *bean* vai ter escopo de Visão (enquanto não trocar de página o *bean* será mantido na memória).



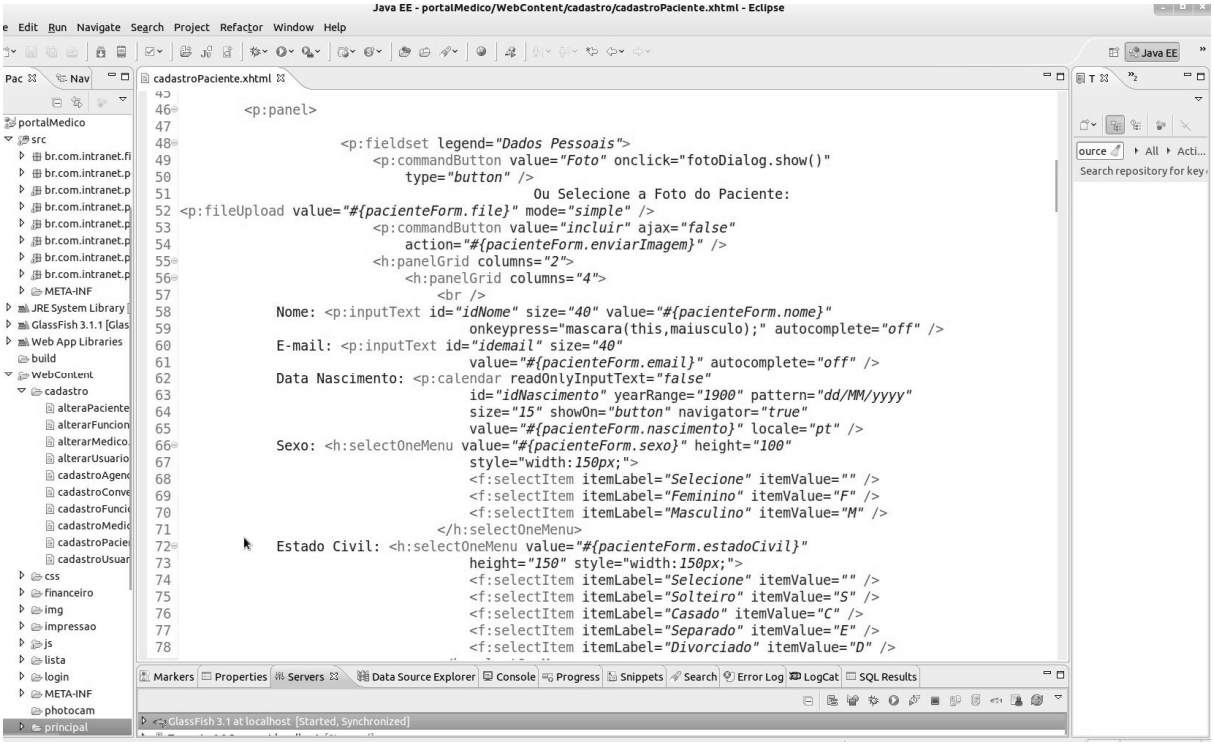
```
1  package br.com.intranet.portalMedico.formulario;
2
3  import java.io.File;
41
42 @ManagedBean(name = "pacienteForm")
43 @ViewScoped
44 public class CadastroPaciente implements Serializable {
45
46     @EJB
47     private PacienteNegocio negocioPaciente;
48
49     @EJB
50     private ConvenioNegocio negocioConvenio;
51
52     @EJB
53     private MedicamentoNegocio negocioMedicamento;
54
55     @EJB
56     private TelefoneNegocio negocioTelefone;
57
58     @ManagedProperty(value = "#{alteraPaci}")
59     private AlterarPaciente beanAlterarPaciente;
```

Markers | Properties | Servers | Data Source Explorer | Snippets | Console | SQL Results | Progress

GlassFish 3.1 at localhost [Started, Synchronized]

Figura 9. Managed Bean com escopo de visão

Uma suíte de componentes JSF customizados conhecido por Prime Faces foi utilizado nas páginas (.xhtml). Conforme pode ser observado na Figura 10. As páginas decidem, por meio de *Expression Language* (EL), quais os dados e as lógicas necessárias para que possam ser processadas.



Java EE - portalMedico/WebContent/cadastro/cadastroPaciente.xhtml - Eclipse

Edit Run Navigate Search Project Refactor Window Help

cadastroPaciente.xhtml

```
46      <p:panel>
47
48          <p:fieldset legend="Dados Pessoais">
49              <p:commandButton value="Foto" onclick="fotoDialog.show()"
50                  type="button" />
51                                  Ou Selecione a Foto do Paciente:
52 <p:fileUpload value="#{pacienteForm.file}" mode="simple" />
53              <p:commandButton value="incluir" ajax="false"
54                  action="#{pacienteForm.enviarImagem}" />
55              <h:panelGrid columns="2">
56                  <h:panelGrid columns="4">
57                      <br />
58      Nome: <p:inputText id="idNome" size="40" value="#{pacienteForm.nome}"
59                          onkeypress="mascara(this,maiusculo);" autocomplete="off" />
60      E-mail: <p:inputText id="idemail" size="40"
61                          value="#{pacienteForm.email}" autocomplete="off" />
62      Data Nascimento: <p:calendar readOnlyInputText="false"
63                          id="idNascimento" yearRange="1900" pattern="dd/MM/yyyy"
64                          size="15" showOn="button" navigator="true"
65                          value="#{pacienteForm.nascimento}" locale="pt" />
66      Sexo: <h:selectOneMenu value="#{pacienteForm.sexo}" height="100"
67                          style="width:150px;">
68                          <f:selectItem itemLabel="Selecione" itemValue="" />
69                          <f:selectItem itemLabel="Feminino" itemValue="F" />
70                          <f:selectItem itemLabel="Masculino" itemValue="M" />
71                      </h:selectOneMenu>
72      Estado Civil: <h:selectOneMenu value="#{pacienteForm.estadoCivil}"
73                          height="150" style="width:150px;">
74                          <f:selectItem itemLabel="Selecione" itemValue="" />
75                          <f:selectItem itemLabel="Solteiro" itemValue="S" />
76                          <f:selectItem itemLabel="Casado" itemValue="C" />
77                          <f:selectItem itemLabel="Separado" itemValue="E" />
78                          <f:selectItem itemLabel="Divorciado" itemValue="D" />
```

Markers | Properties | Servers | Data Source Explorer | Console | Progress | Snippets | Search | Error Log | LogCat | SQL Results

GlassFish 3.1 at localhost [Started, Synchronized]

Figura 10. Página (.xhtml) utilizando JSF e Prime Faces

Na Figura 11 é apresentada a tela do sistema em funcionamento. A agenda de pacientes está em destaque para fornecer acesso rápido aos pacientes agendados. Por meio do menu lateral é possível acessar todas as funcionalidades da aplicação, desde os cadastros de

funcionários, pacientes, convênios, até o controle financeiro, cadastro e controle de consultas.

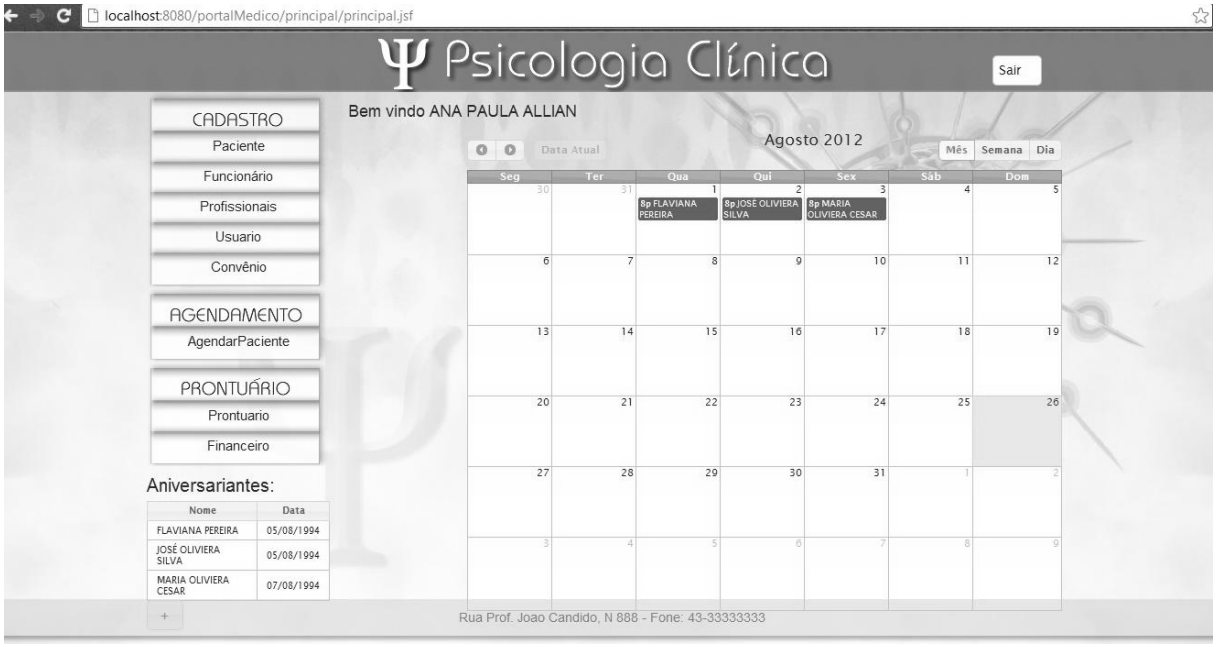


Figura 11. Tela Principal do Sistema em Funcionamento

4 LIÇÕES APRENDIDAS

4.1 IMPLEMENTAÇÃO DA INTERFACE GRÁFICA

Apesar de se ter utilizado um perfil reduzido para aplicações Web, no sistema proposto foi utilizado apenas JSF. O JSF é um *framework* para simplificar a integração de desenvolvimento de interfaces baseadas na Web. JSF tem um ciclo de vida mais complexo em que a geração do componente e a sua renderização acontecem em fases claramente separadas. O JSP é uma API com funcionalidades semelhantes ao JSF, porém menos flexível e é usado para criar conteúdo estático ou dinâmico na Web, seus elementos são processados com o objetivo fundamental de criar uma resposta a um pedido. Também não foi utilizado o *BeanValidation* pois, nesse projeto, as validações e as regras de negócio foram implementadas na camada de negócio da aplicação.

4.2 ADOÇÃO DA VERSÃO EJB LITE

Observando as funcionalidades do EJB *Lite* da Tabela 1, nota-se que também foram descartadas algumas APIs do EJB *Lite*. Neste projeto foi utilizado apenas o novo tipo de *bean* de sessão introduzido na especificação Java EE 6 conhecido por *Singleton*. Uma única instância de cada *bean* de sessão *Singleton* é criada no servidor de aplicativos. Assim, constatou-se que os *beans* da sessão *Singleton* são úteis para fazer o *cache* de banco de dados. A grande vantagem do *Singleton* é que o *cache* frequentemente usa os dados de um *bean* de sessão, aumentando o desempenho da persistência, uma vez que minimiza bastante os acessos às bases de dados (Heffelfinger, 2010).

4.3 FUNCIONALIDADES DO SISTEMA

As funcionalidades do sistema são tarefas que o usuário poderá executar através do sistema (consulta, cadastro, controle financeiro, etc). É importante ressaltar que todas essas

tarefas são facilmente executáveis através do menu presente na tela principal do sistema. Dentre as funcionalidades do sistema desenvolvido destacam-se:

- Cadastro de Pacientes, funcionários, usuários e profissionais;
- Registro de pacientes com fotos através de webcam;
- Agenda de pacientes;
- Controle financeiro;
- Exportar dados para planilha eletrônica;
- Impressão de atestado e declarações;
- Registro diário de consultas;
- Controle de permissão do sistema através de cadastro de usuário por perfil;

5 CONCLUSÕES E TRABALHOS FUTUROS

Este artigo apresentou uma experiência de desenvolvimento de um sistema Web para gerenciamento de clínicas de psicologia, utilizando a tecnologia *Web Profile* do Java EE 6. Tal tecnologia engloba outras tecnologias e ferramentas para facilitar e agilizar o desenvolvimento de aplicações Web de pequeno e médio porte.

O sistema Web foi desenvolvido com o objetivo de fornecer uma abordagem prática, na construção de aplicações Web *lightweight*, fazendo uso das APIs do *Web Profile*, evidenciando na prática que essas ferramentas facilitam o desenvolvimento de sistemas com maior agilidade. Todo o suporte necessário para compreender a rotina do consultório de psicologia foi fornecido pelos psicólogos que disponibilizaram informações imprescindíveis para a criação das regras de negócio e o desenvolvimento do sistema.

Por simplicidade, o Web *Profile* deixa de fora muitas das APIs corporativas de *backend* que são parte da plataforma Java EE 6 e, em contrapartida, os sistemas que utilizam Web *Profile* podem ser entregues com mais APIs do que as exigidas. A aplicação desenvolvida nesse projeto não necessita de APIs externas ao *Web Profile* porém, caso necessite, é necessária a inclusão da mesma por meio da ferramenta de *Update Tool* no servidor *GlassFish*. É possível adicionar quantas APIs forem necessárias, resultando em um perfil novo e personalizado, porém não mais no *Web Profile*.

Como direção para possíveis trabalhos futuros tem-se: (i) a necessidade de desenvolver testes de unidade e de desempenho para o software desenvolvido com a tecnologia *Web Profile*, comprovando seu custo computacional; (ii) continuar buscando o aprimoramento do sistema, incluindo novas funcionalidades na forma de *Web services* para a integração com dispositivos móveis visando agendar consultas, por exemplo; e (iii) estender o desenvolvimento à medida que novas tecnologias e ferramentas forem disponibilizadas no mercado.

REFERÊNCIAS

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML Guia do Usuário. 2ed. Rio de Janeiro RJ: Elsevier, 2006.

ECLIPSE. IDE Eclipse: Disponível em: <http://www.eclipse.org/>. Acesso em: 01, Set, 2012.

GEARY, David; HORTSMANN, Cay S. Core Java Server Faces. 3ed. EUA: Prentice Hall, 2010.

GLASSFISH. Servidor GlassFish: Disponível em: <http://glassfish.java.net/>. Acesso em: 01, Set, 2012.

GONÇALVES, Antônio. Introdução à Plataforma Java EE6 com GlassFish. 2ed. Rio de Janeiro RJ: Ciência Moderna, 2011.

HALL, Marta; BROWN, Larry. Core Servlets and Java Server Pages. 2ed. EUA: Prentice Hall and Sun Microsystems Press, 2000.
HEFFELFINGER, David. Java EE6 with GlassFish 3 Application Server. 2ed. Birminghan UK: Packt Publishing, 2010.
JAVA COMMUNITY. Java Community Process. Disponível em: <http://www.jcp.org/en/home/index>. Acesso em: 04, Out, 2012.
MOLINARI, Luciano. Java EE 6 da configuração aos Testes. Java Magazine, ed. 106, p.28 – 40, 2012.
MySql - MySQL 5.5 Reference Manual: Disponível em: <http://dev.mysql.com/doc/refman/5.5/en/introduction.html>. Acesso em: 26, Set, 2012.
NIEDERAUER, Juliano. MySQL 5. Guia de Consulta Rápida. 2ed. São Paulo : Novatec, 2006.
ORACLE. O que há de novo na versão do Sun GlassFish Enterprise Server v3?: Disponível em: <http://docs.oracle.com/cd/E19226-01/821-1337/ggpnv/index.html>. Acesso em: 26, Set, 2012.
ORT, Ed. Introducing the Java EE 6 Platform: Part 1: Disponível em: <http://www.oracle.com/technetwork/articles/javaee/javaee6overview-part3-139660.html>. Acesso em: 01, set, 2012.
PANDA, Debu, RAHMAN, Reza; LANE, Derek S. EJB 3 In Action. 3.ed. Greenwich, CT : Manning, 2007.
RAHMAN, Reza. Introducing the Java EE Web Profile. Disponível em: <http://jaxenter.com/introducing-the-java-ee-web-profile-36201.html>. Acesso em: 15, Set, 2012.
RUBINGER, Andrew Lee; BURKE, Bill. Enterprise Java Beans 3.1. Sebastopol, CA – EUA: O' Reilly Media Inc, 2010.
SAMPAIO, Cleuton. Java Enterprise Edition 6. Desenvolvendo Aplicações Corporativas. 2ed Rio de Janeiro : Brasport, 2011.
SOUZA, Vitor E. S. Java EE 6 Web Profile. Java Magazine, ed. 100, p.12 – 25, 2012.
SOUZA, Vitor E. S. Java EE 6 na Prática. Java Magazine, ed. 80, p. 20 – 34, 2010.