Histórico da Revisão

Data	Versão	Descrição	Autor

Sumário

Documento de Arquitetura

1. Introdução

1.1 Propósito

Este documento apresenta a arquitetura proposta para o sistema SG-DRCI. Tem como objetivo capturar e conduzir as decisões arquiteturais a serem tomadas.

As seguintes visões serão abordadas:

- Visão de Caso de Uso;
- Visão Lógica;
- Visão de Processos;
- Visão de Implementação;
- Visão de Implantação.

1.2 Glossário

Termo	Significado
Ant	Utilitário escrito em Java utilizado para a automatização de tarefas de compilação e geração de pacotes para a plataforma Java
Arquivo de propriedades	Arquivos associam uma chave a valor específico com objetivo de retirar do código-fonte utilização direta desse valor, facilitando a alteração em casos de possíveis manutenções.
Biblioteca	Conjunto de funcionalidades Java reutilizáveis dentro de um sistema.
Browser	Um aplicativo para acesso a documentos, normalmente em formato HTML que residem em um servidor WEB.
CGTI	Coordenação-Geral de Tecnologia da Informação.
Container	Engine do servidor de aplicação
Controle de versão	A gestão de diferentes versões no desenvolvimento de um documento qualquer efetuada por um software específico que atua como servidor e suas aplicações cliente.
CMT	Container Managed Transaction – (Transação Gerenciada pelo Container) Essa opção faz com que o próprio container faça o controle do início e final da transação em um método de negócio da aplicação.
CSS	Cascade Style Sheets, para padronização de estilos.
Data source	Mecanismo que cria conexões com banco de dados gerenciados pelo servidor de aplicações.
Design Pattern	Solução de projeto de implementação desenvolvida por terceiros, que pode ser reaproveitada em vários projetos. Não confundir com componente reutilizável; o design pattern não é fornecido com implementação
EJB	Componente negocial JAVA corporativo, Enterprise Java Beans. Os principais objetivos da tecnologia EJB são fornecer um rápido e simplificado desenvolvimento de aplicações Java baseado em componentes distribuídas, transacionais, seguras e portáveis
Fachada	Design Pattern para definição de uma interface mínima de comunicação entre um subsistema cliente e um fornecedor de serviços
IDE	(Integrated Development Environment) Ambiente Integrado de Desenvolvimento. Software que reúne características e ferramentas de apoio ao desenvolvimento de sistemas com o objetivo de agilizar este processo.
Hibernate	É um framework para o mapeamento objeto-relacional escrito na linguagem Java.

HTML	Hyper Text Makup Language. Linguagem de marcação utilizada para construir páginas web.
JAR	Arquivos que encapsulam componentes ou bibliotecas Java.
JDBC	Java Database Connectivity é um conjunto de classes e interfaces (API) escritas em Java que faz o envio de instruções SQL para qualquer banco de dados relacional
JPQL	Java Persistence Query Language, bem similar ao SQL mas orientado a objetos.
Lookup	Ação pela qual um sistema faz a chamada remota a um componente EJB.
Model View Controler	MVC: É o design pattern utilizado na camada de apresentação
Modelo de Projeto	Conjunto de artefatos relacionados com a disciplina de análise e design, que complementam os artefatos da disciplina de requisitos e que serão utilizados para guiar a construção de componentes
Padrão de projeto	Ver Design Pattern
POJO	Plain Old Java Object. Objeto Java que segue um desenho simplificado em contraposição à complexidade de um componente EJB
Segurança declarativa	Cada recurso pode ser acessado por uma ou mais roles. O usuário pode pertencer a uma ou mais roles, ou seja, pode ter um ou mais papeis dentre um aplicativo. Isso é equivalente a estrutura organizacional de uma empresa.
Segurança programática	Cada recurso que pode ser acessado por uma ou mais roles é controlado pela aplicação via código específico que acessa o mecanismo de segurança diretamente.
SGBD	Sistema de Gerenciamento de Banco de Dados
Taglib	Uma biblioteca a ser usada dentro de arquivos JSP.
Thread	É uma linha de execução efetuada pelo processador da máquina.
UML	Unified Modeling Language
Value Objecs	VO: Design pattern utilizado para transferência de objetos entre as camadas da aplicação
WAR	Arquivos que encapsulam sistema web desenvolvidos em Java.
WEB	Sistema baseados em HTML interligados e executados por meio de um browser.
XML	eXtensible Markup Language

1.3 Referências

- Visão e requisitos: Sixxx_DVR.doc
- Regras de negócio: Sixxx_RN.doc
- Core J2EE Patterns, Deepak Alur, Jhon Crupi, Dan Malks; Campus, 2002;
- Code Conventions for the Java Programming Language - http://java.sun.com/docs/codeconv;
- The 4+1 view model of architecture, IEEE Software. 12(6), November 1995. Kruchten, P.;
- Documentação Javadoc – http://java.sun.com/j2se/javadoc;
- Hibernate - http://www.hibernate.org;

2. Representação da arquitetura

A arquitetura definida para o desenvolvimento do projeto é baseada na plataforma Java EE. Esta plataforma estabelece padrões para construção de componentes de software, baseados em camadas com responsabilidades e recursos específicos. Java EE é baseado numa padronização modular de componentes com um conjunto de API padrão que traz comportamentos da aplicação já tratados, permitindo que os desenvolvedores foquem na criação de componentes para tratar de detalhes da aplicação diminuindo a complexidade no desenvolvimento. A unificação de uma plataforma proporciona um rápido desenvolvimento, maior manutenibilidade, reusabilidade dos componentes e portabilidade na aplicação.

2.1 Modelo de visão 4 + 1

Como forma de abordar um determinado conjunto de questões específicas dos envolvidos no processo de desenvolvimento, optou-se por representar a arquitetura de software em várias visões de arquitetura. Estas visões, em sua essência, são fragmentos que ilustram os elementos significativos em termos de arquitetura dos modelos. Partiremos de um conjunto típico de visões denominado "modelo de visão 4 + 1" proposto por Philippe Kruchten.

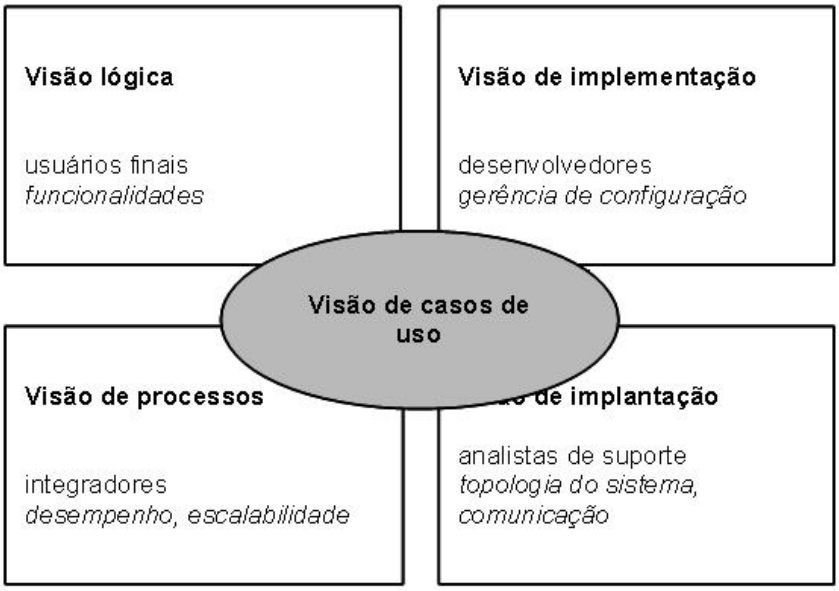

Ilustração 1: O modelo de visão 4 + 1

Cada uma dessas visões serão tratadas em seções específicas deste documento.

3. Composição da Arquitetura

3.1 Tecnologias

Tabela 1 Tecnologias utilizadas

Nome	Versão	Descrição	Utilização
EJB	3	Ou Enterprise JavaBean, é componente executado no container do servidor de aplicação. Tem a função de prover a infraestrutura necessária para a execução dos componentes de negócio da aplicação.	Utilizada na camada de negócio
J2SE	1.5	É uma plataforma de desenvolvimento para a linguagem Java. Composto por uma máquina virtual e um conjunto de bibliotecas necessárias para acesso a arquivos, redes e o uso de interfaces gráficas, entre outros	Utilizada em toda a aplicação
J2EE	1.5	Plataforma para a construção de aplicações servidoras distribuídas, multi-camadas, transacionais e modulares.	Toda a aplicação é baseada nessa plataforma
JNDI		API para acesso a serviços de diretórios. Ela permite que aplicações cliente descubram e	Utilizada na camada de negócio e dados

		obtenham dados ou objetos através de um nome.	
Annotation		Recurso da Plataforma Java que fornece a opção do uso de metadata ao longo do código que podem ser posteriormente interpretadas por um compilador ou pré-compilador que irá realizar alguma tarefa pré-definida.	Utilizada em toda as camadas
MS SQL Server	2005	Servidor de banco de dados	SGBD utilizado como o repositório de dados.
JSP		Tecnologia utilizada no desenvolvimento de aplicações para Web.	Visão
JSTL	1.1	JavaServer Pages Standard Tag Library, é um componente da plataforma de desenvolvimento web Java EE. Ela estende a especificação JSP adicionando uma biblioteca de tags das tags JSP para tarefas comuns, tais como processamento de dados XML, execução condicional, loops e internacionalização.	Visão
JPA	2	Especificação para acesso à camada de dados.	Utilizada na camada de dados

3.2 Ferramentas

Tabela 2 Ferramentas

Nome	Versão	Descrição	Utilização
Java SDK	1.6.24	Também chamado de JDK. Ela contém todo o ambiente necessário para a criação e execução de aplicações Java, incluindo a máquina virtual Java (JVM), o compilador Java, as APIs do Java e outras ferramentas utilitárias.	Permite a construção e execução de software baseado a plataforma Java.
Eclipse	3.0 ou superior	Um ambiente de desenvolvimento (IDE), composto por um editor e plugins.	Editor de código, automatização de tarefas, geração de código e auxílio a depuração de código, entre outros.
Internet Explorer	7	É um navegador de Internet de licença proprietária produzido pela	Acesso a documentos, normalmente em formato html que residem em um servidor

		Microsoft.	WEB.
Firefox	7	É um navegador de Internet livre e multi-plataforma desenvolvido pela Mozilla Foundation.	Acesso a documentos, normalmente em formato html que residem em um servidor WEB.
TortoiseSVN	1.6.6 ou superior	É um cliente com código aberto do Subversion para Microsoft Windows	Controle de versão de código
Ireport	1.1	Um aplicativo utilitário Java para a construção de relatórios	Gerar os templates XML de relatórios.
Ant	1.1.4 ou superior	Utilitário escrito em Java para a automatização de tarefas	Utilizado para a automatização de tarefas de compilação e geração de pacotes para a plataforma Java

3.3 Frameworks

Tabela 3 Frameworks utilizados

Nome	Versão	Descrição	Utilização
Hibernate	3.2	O Hibernate é um framework Open Source para o mapeamento objeto-relacional escrito na linguagem Java. Este framework facilita o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação, mediante o uso de Annotations para estabelecer esta relação.	Transformação das classes em Java para tabelas de dados.
JasperReports	4.0.1	Framework Java para a geração de relatórios em diversos formatos como PDF e XSL.	Todos os relatórios gerados estarão no formato PDF
JSF	2	Framework Java que implementa o padrão de projeto MVC para ser utilizado em um servidor Java EE.	Controle e Visão
Primefaces	2.2.1	Framework MVC extensão do JSF	Visão

3.4 Componentes corporativos

Tabela 4 Componentes corporativos

Nome	Versão	Descrição	Utilização
Corporativo	1.0	Componente corporativo composto de diversas classes utilitárias e centralizador do acesso a dados corporativos, entre outros.	Acesso a dados corporativos
Seguranca2	1.0	Componente corporativo centralizador para a utilização de autenticação, autorização e auditoria	Autenticação, autorização e auditoria

Core3	1.0	Componente corporativo composto de diversas classes base para outros componentes	Classes utilitárias diversas
Util	1.0	Componente corporativo composto de diversas classes utilitárias	Classes utilitárias diversas
Processo	1.0	Componente do framework de processos	Acesso a documentos e/ou protocolos
CoreJSF	1.0	Componente corporativo composto de diversas classes utilitárias para suporte de funções JSF.	Classes utilitárias diversas.
CoreSG	1.0	Componente corporativo composto de diversas classes utilitárias.	Classes utilitárias diversas.
CoreReport	1.0	Componente corporativo composto de diversas classes de suporte a relatórios.	Classes utilitárias diversas.

3.5 Padrões de Projeto

Tabela 5 Descrição dos padrões de projeto utilizados

Nome	Utilização
MVC - Model View Controller	Direcionar a construção da interface gráfica da camada de apresentação.
Facade	Expor os serviços da camada de negócio
DAO – Data Acess Object	Todo o acesso à base de dados está centralizada em classes que implementam esse padrão
Business Delegate	Agrupar o acesso aos componentes de negócio
Singleton	A classe utilitária para a geração de log é baseada nesse padrão
Value Object	Todas as classes de filtro de dados estão baseadas nesse padrão. As classes POJO são tratadas como Value Object quando não há a necessidade de tratamento para dados compostos ou complementares
Service Locator	Utilizado para a localização dos componentes de negócio
Command	No sistema essa separação corresponde aos ManagedBean do JSF

4. Requisitos e restrições arquiteturais

4.1 Tecnologias de desenvolvimento

- A arquitetura da aplicação deve ser baseada na arquitetura Java EE 1.5;
- O sistema deverá ser executado como uma aplicação WEB;
- Para implementação da camada de negócio será utilizado tecnologia EJB 3;

- As versões de componentes e frameworks são limitados ao conjunto homologado pela CGTI.

4.2 Servidor de Aplicação

- O servidor de aplicação a ser utilizando é o JBoss, versão 4.2.3 ou superior.

4.3 Servidor de banco de dados

- Deve ser utilizado o MS SQLServer como servidor de banco de Dados (SGBD), versão 2005.

4.4 Protocolo de rede

- O acesso ao sistema deverá ser realizado através da intranet (utilizando protocolo HTTP). A invocação dos componentes de negócio será feita via RMI.

4.5 Padrões corporativos

- O layout das janelas de diálogo e o uso de componentes deve seguir os padrões definidos pela CGTI;
- O sistema deverá possuir interface baseada em WEB;
- Para função de log de transações será reutilizado componente disponibilizado pela CGTI.

4.6 Segurança

- A segurança na aplicação será baseada na segurança declarativa: autenticação e autorização de usuário (login / senha) via componente de segurança da aplicação;
- Para auditoria das ações dos usuários, o sistema utilizará o componente corporativo de segurança para o registro de log.

4.7 Ambiente de desenvolvimento

- Codificação utilizando a IDE Eclipse;
- Linguagem Java;
- As versões de componentes e frameworks são limitados ao conjunto homologado pela CGTI.

5. Camadas

A arquitetura proposta, segundo o modelo Java EE, está estruturada em camadas independentes e sobrepostas. Pode-se dizer que uma camada poderá agrupar classes, pacotes e outros componentes que possuam funcionalidades em comum.

- Camada de visão. Essa camada é responsável por toda a interface com o usuário. Ela intercepta requisições do cliente, gerencia a sessão, controla acesso aos serviços de negócio, constrói as respostas e entrega aos clientes. A camada cliente será implementada com padrões WEB por páginas JSP, HTML, XHTML e arquivos CSS entre outros.
- Camada de controle. Essa camada determina o fluxo da apresentação servindo como uma camada intermediária entre a camada de apresentação e a de negócio. A lógica de apresentação será implementada utilizando-se o padrão MVC (Model View Controller). A comunicação entre esta camada e a camada de regra de negócio é realizada através da implementação do padrão Business Delegate.
- Camada de negócio. Esta camada contém componentes responsáveis pelas regras de negócio do sistema, ou seja, ela fornece os serviços de domínio da solução. Esses serviços são disponibilizados através de uma interface pública implementada pela utilização do padrão de projeto Session Facade. O acesso a estes serviços sempre é disponibilizado através de classes que utilizam o padrão de projeto Business Delegate. Para realizar a busca destes componentes de serviços é necessário realizar uma tarefa chamada lookup. Para isto será utilizado o padrão de projeto chamado Service Locator.

- Camada de dados. A camada de persistência possui componentes responsáveis pela lógica inerente à persistência e recuperação de informações desde o sistema de armazenamento. O objetivo dessa camada é fornecer uma abstração capaz de encapsular toda a lógica de acesso a dados, deixando transparente a camada de negócio como são realizadas tais operações. O tipo de sistema de armazenamento (base de dados, XML, sistemas legados e etc) utilizado seria de conhecimento exclusivo da camada de persistência, sendo irrelevante para as outras camadas da arquitetura. Isso permite que o sistema de armazenamento seja alterado (XML para banco de dados, por exemplo), sem alterações nas demais camadas da aplicação, o que facilitaria, em muito, uma possível manutenção nesse sentido. O padrão utilizado para se conseguir tal objetivo é o Data Access Object (DAO).

Ilustração 2 Visão geral da arquitetura em camadas

As camadas acima são detalhas nas demais sessões do documento.

6. Visão de caso de uso

Os casos de uso apresentados abaixo representam uma funcionalidade central e significativa do sistema final ou possuem uma ampla cobertura de arquitetura, ou seja, experimentam muitos elementos arquiteturais ou que enfatizam ou ilustram um determinado ponto frágil da arquitetura.

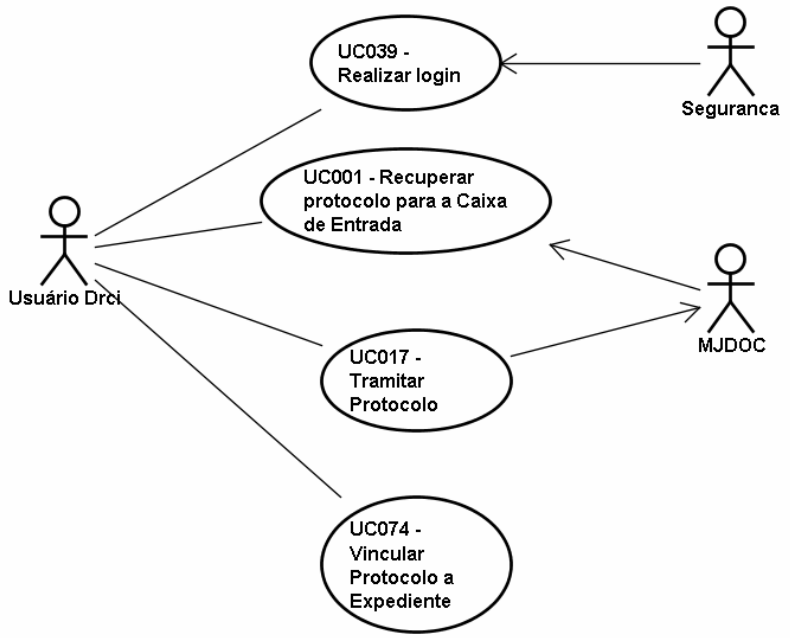

Ilustração 3: Casos de uso significativos

7. Visão lógica

7.1 Visão geral

O Sistema será projetado e implementado através de uma estrutura de módulos independentes e particionados tanto física quanto logicamente. Utilizará o modelo MVC mais uma camada específica para acesso a dados persistentes, sendo:

- Modelo: Classes de negócio ou classes utilizadas por estas. Será composta pelas classes EJB, classes de entidades (pojo), classes utilitárias (VO e utils) e as classes de negócio;
- Visão: Classes ou arquivos para interação com o usuário. Será composto por arquivos: jsp, HTML e XML;
- Controle: Classes que fazem o papel de integrar a camada de visão com a camada de modelo. Será composta por classes utilitárias (VO e utils) e classes de Controller;
- Persistência: Classes que abstraem o acesso a dados persistentes. Será composta pelas classes de DAO.

Fisicamente, a aplicação estará decomposta nos pacotes WEB e Componente. Assim, pressupõe-se que toda aplicação possuirá ou acessará um componente que conterá toda a regra de negócio do sistema não existindo implementações negociais nas demais partes do sistema. Esse acesso ao componente acontecerá de forma direta pela camada de controle e essa, por sua vez, poderá ser implementada na parte e na parte WEB da aplicação.

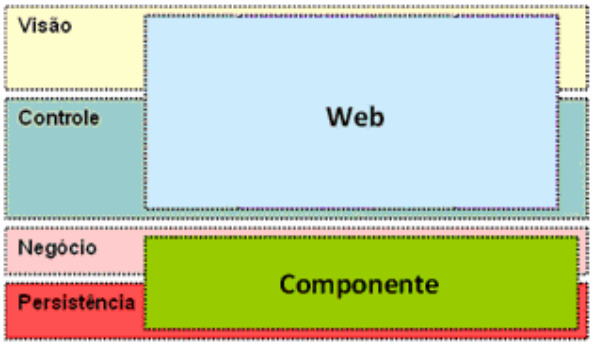

Ilustração 4 Camadas da aplicação

7.2 Pacotes arquiteturalmente significativos

A aplicação possui os pacotes listados na tabela abaixo que são significativos do ponto de vista arquitetural.

Dos módulos a serem criados para compor o sistema, o mais importante será o módulo EJB, uma vez que esse módulo encapsulará toda a regra de negocio envolvida no sistema, dessa forma a aplicação web não será nada mais que uma interface para interação com o usuário.

Tabela 6 Pacotes significativos arquiteturalmente

WEBDrci	br.gov.drci.mb	Representam a camada de controle da aplicação WEB, contendo a lógica de recuperação e de envio de valores da interface gráfica bem como a lógica de navegação.
COMPDrci	br.gov.drci.ejb	Contém as classes de negócio e suas fachadas.
	br.gov.drci.servico	Contém as interfaces de serviços da aplicação
	br.gov.drci.pojo e br.gov.drci.dao	Responsável por armazenar as entidades e classes responsáveis pelo acesso à base de dados relacional
	br.gov.drci.util	Concentra as classes utilitárias e de suporte

7.3 Visão de implementação

7.3.1 Visão geral

A aplicação será dividida em dois grandes conjuntos: um para o componente que atenderá a parte negocial e persistência, e outro para a aplicação cliente. Cada pacote será implementado como um projeto Eclipse seguindo o wizard Java Project.

Não são permitidos a utilização de classes da aplicação web ou standalone na camada EJB nem a referência entre projetos do Eclipse.

Os pacotes serão definidos como a seguir:

- COMPDrci: componente com a parte negocial e de persistência da solução, possui um diretório src que conterá os pacotes e classes Java;
- WEBDrci: agrupa os objetos das camadas de apresentação e controle. Dividido em dois diretórios:
    - src. Conterá os pacotes e classes Java.
    - WEBDrci.war. Contém os elementos visuais. O sub-diretório Webcontent agrupa as páginas JSP, arquivos css, imagens e código Javascript. O sub-diretório WEB-INF contém os arquivos descritores e de configuração da aplicação.

O Ant será utilizado para a geração dos componentes de implantação e utilizará, em cada um dos projetos Eclipse, os seguintes arquivos de definição de tarefas, já aderentes aos arquivos XML corporativos:

- COMPDrci:
    - build.properties
    - build.xml

A utilização de nomes de variáveis, de classes e demais objetos Java e/ou modelo J2EE devem seguir os padrões definidos pelas SUN e já adotados pela comunidade de desenvolvedores Java. (Ver a seção 1.3)

A comunicação entre as camadas será implementado com o uso dos padrões de projeto Value Object e POJO.

7.3.2 Aplicação WEB

O acesso ao sistema pela WEB será realizado através do protocolo HTTP, com a utilização de navegadores no cliente e a implementação da camada de controle e visão será realizada pelo JSF, versão 2, que representará as camadas de controle de visão do modelo MVC.

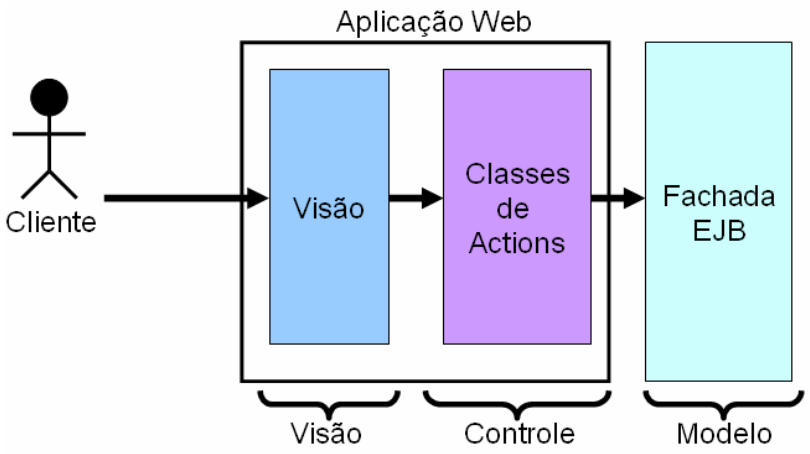

Ilustração 5 Comunicação entre as camadas da aplicação WEB

Existirão dependências do componente apenas com outros componentes que compõe a solução corporativa ou bibliotecas que existam no diretório /java/Biblioteca com exceção do diretório corporativo. Não sendo possível a utilização de classes da aplicação web no componente EJB.

Será utilizado o framework Primefaces como a implementação MVC JSF.

Tabela 7 Estrutura de pacotes da camada de apresentação

Pacote	***Descrição***
br.gov.drci.mb	Classes ManagedBean para controle (Controller).
br.gov.drci.util	Classes utilitárias que serão utilizadas pela camada de apresentação.
pagina	Contém as páginas jsp, html e xhtml

7.3.2.1 Camada de controle

Para compor a camada de controle, para cada caso de uso do sistema será criado uma classe ManagedBean do JSF que conterá todas as chamadas ao componente para atender ao caso de uso em questão. Ela será responsável pelo intercâmbio dos dados entre as camadas de visão e negocio bem como realizar as devidas validações de campos quando necessário.

Por se tratar da arquitetura MVC e o JSF representar a camada de visão e controle, não deverá existir nenhum tipo de acesso ou redirecionamento de URL que não seja controlado pelo JSF, ou seja, não deverá existir redirecionamento direto entre arquivos (utilização da tag HTML <a href=""> apontando diretamente para arquivos físicos) ou a utilização do comando sendRedirect para redirecionamento entre as actions e outras partes de um mesmo sistema. Todo o controle de navegação estará centralizado no arquivo faces-config.xml

7.3.2.1.1 Validação

A utilização de validações nessa camada não desobriga a validação no lado servidor.

7.3.2.2 Camada de visão

Essa camada será composta por arquivos JSP, XML, XHTML e HTML e implementadas sob o framework PrimeFaces.

Para a formatação das páginas, será feita a utilização de arquivos CSS.

As páginas estarão agrupadas por caso de uso.

7.3.2.2.1 Arquivo de mensagens

Toda aplicação web possuirá um arquivo de propriedade que centralizar as mensagens e rótulos do sistema, cada um deles estará associado a uma chave única dentro desse arquivo e o sistema só utilizará essas chaves.

Para utilização do arquivo de propriedade é necessário configurar o faces-config.xml conforme exemplo abaixo:

```
<application>
    <resource-bundle>
        <base-name>br.gov.drci.resources</base-name>
        <var>rotulo</var>
    </resource-bundle>
    <locale-config>
        <default-locale>pt</default-locale>
        <supported-locale>en</supported-locale>
```

```
    </locale-config>
</application>
```

7.3.2.2.2 Contexto da aplicação

O nome do contexto da aplicação (context-root) será especificado no arquivo WEB-INF/jboss-web.xml, assim como a forma repasse das requisições de classes ao container (java2ParentDelegation) sempre definido com false e o classloader da aplicação (loader-repository) que deverá ser associado ao WAR gerado para o sistema, ver exemplo número 5.

Não será utilização o nome do arquivo WAR para definir o nome do contexto da aplicação.

7.3.2.3 Segurança

A proteção de recursos é baseada em roles ou papéis, conforme definido pela segurança declarativa no padrão J2EE.

A proteção dos recursos é baseada na URL sendo invocada (correspondente a um dos pacotes de implementam os casos de uso) e especificada no deployment descriptor da aplicação (web.xml), na seção security-constraint.

```
    <security-constraint>
        <display-name>Cadastros</display-name>
        <web-resource-collection>
            <web-resource-name>Suporte</web-resource-name>
            <url-pattern>/comum/*</url-pattern>
            <url-pattern>/Webcontent/paginas/manterinstituicao/*</url-
pattern>
            <url-pattern>/mantercategoria/*</url-pattern>
            <url-pattern>/Webcontent/paginas/mantercategoria/*</url-
pattern>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
                  <role-name>2214</role-name>
                  <role-name>2215</role-name>
                  <role-name>2216</role-name>
                  <role-name>2217</role-name>
                  <role-name>2218</role-name>
        </auth-constraint>
        <user-data-constraint>
            <transport-guarantee>NONE</transport-guarantee>
        </user-data-constraint>
         </security-constraint>
```

Exemplo de código 1 Especificação de recurso WEB protegido

O autenticador utilizado deve ser o realm, loginCorp, declarado do deployment descriptor para o servidor de aplicação utilizado (jboss-web.xml)

```
<jboss-web>
    <context-root>/drci</context-root>
    <class-loading java2ClassLoadingCompliance="false">
        <loader-repository>
            br.gov.drci:loader=WEBDrci.war
            <loader-repository-config>java2ParentDelegation=false</loader-
repository-config>
        </loader-repository>
    </class-loading>
    <security-domain
flushOnSessionInvalidation="true">java:/jaas/loginCorp</security-domain>
</jboss-web>
```

Exemplo de código 2 jboss-web.xml

7.3.2.4 Tratamento de exeções

Exceções do tipo br.gov.core.exception.NegocioException indicam que a exceção faz parte do fluxo normal da aplicação, por isso o comportamento a ser adotado pelo tratador será encaminhar o usuário para a página definida no atributo input da tag action no arquivo de configuração do JSF e exibir a mensagem de erro contida na exceção.

Para exceções que não representarem o fluxo normal da aplicação (java.lang.Exception ou br.gov.core.servico.exception.ErroInternoException) o comportamento do tratador de exceções será de encaminhar o usuário para página de erro padrão do sistema.

7.3.3 Componente

O componente engloba toda parte negocial da aplicação e a parte persistência de dados. Corresponde a camada de "Modelo" do modelo MVC e a camada de persistência.

O componente será construído usando EJB e contemplará as classes de VO, classes utilitárias e persistência de dados (DAO e POJO).

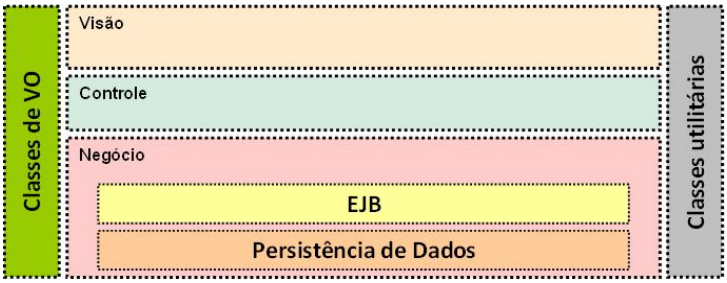

Ilustração 6 Camadas de um componente

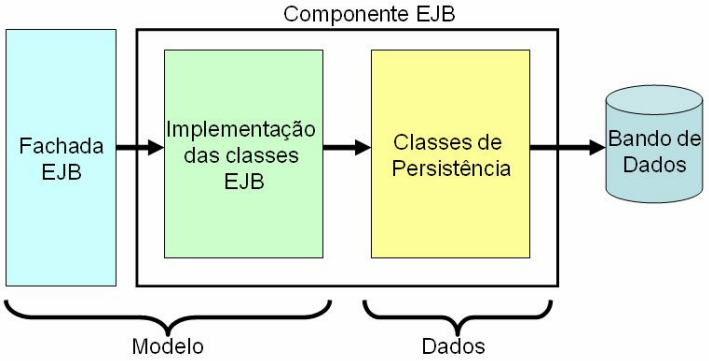

Ilustração 7 Comunicação entre as camadas do componente

Todas as classes do componente serão agrupadas em subdiretórios no diretório src. Cada um desses subdiretórios será um pacote da aplicação. Eles serão divididos da seguinte forma:

Tabela 8 Estrutura de pacotes do componente:

Pacote	***Descrição***
br.gov.drci.ejb	EJBs de sessão stateless e classes business delegate responsável por encapsular a lógica de localização do EJB de Fachada.
br.gov.drci.util	Classes de utilidade que serão utilizadas por todo o projeto
br.gov.drci.exception	Classes de exceção personalizadas para a aplicação. Somente exceções referentes à camada negócio estarão neste pacote

Pacote	***Descrição***
br.gov.drci.servicos	Intefaces que definem os serviços que conterão as regras de negócio. Tanto o BusinessDelegate quanto o EJB de negócio (Fachada de negócio de sessão) deverão implementar esta interface afim de que sejam obrigados a terem os mesmos métodos de negócio
br.gov.drci.pojo	Classes de entidades representando o modelo objeto relacional (ORM)
br.gov.drci.bo	Classes com as regras de negócio.
br.gov.drci.vo	Conterá utilizadas apenas para transporte de dados entre as camadas sem possuir nenhum tipo de inteligência
br.gov.drci.dao	Classes DAO e DAOFactory que encapsulam o acesso ao SGDB

7.3.3.1 EJB

As classes EJB, nomeadas como *ClasseXXXFacade*, serão fachadas para os componentes de negócio, que acionarão as classes DAO apropriadas a seu caso. O EJB instanciará um EntityManager JPA que será repassado ao componente de negócio no seu construtor.

O serviço provido pelo EJB será acessado por meio de classe que utilizará o pattern Business Delegate e encapsulará todo processo de lockup ao EJB. Seu construtor fará lookup no arquivo jndi.properties.

Para exceções às regras negociais podem ser criadas exceções especificas para cada não conformidade com essas regras. Essas exceções herdarão da classe br.gov.core.exception.NegocioException contida na biblioteca Core.jar.

Todas as ações realizadas pelo usuário dentro do componente EJB deverão realizar log de auditoria de dados por meio do componente de log do corparativo.

Tabela 9 Nomes JNDI das fachadas EJB

Nome JNDI	***Descrição***
br/gov/drci/ejb/DrciSuporteFacade	Acesso aos serviços referentes a tabelas auxiliares e classes utilitárias
br/gov/drci/ejb/DrciFacade	Acesso aos serviços relativos aos processos do DRCI

7.3.3.2 Camada de persistência

Em uma aplicação onde o sistema de armazenamento é uma base relacional, os objetos de dados são o resultado do mapeamento objeto/relacional que torna possível que o sistema seja desenvolvido seguindo o paradigma OO. A implementação deste mapeamento é efetuado na aplicação por anotações JPA.

A camada de persistência é implementada com o Design Pattern DAO e baseada em JPA. É responsável por armazenar as entidades e classes responsáveis pelo acesso a base de dados relacional.

Toda parte transacional é delegada ao container não sendo feito nenhum tipo de tratamento no componente.

Todas as classes DAO estenderão a classe `br.gov.core.dao.AbstractJPADAO` contida na biblioteca Core.jar.

As classes DAO serão obtidas por meio uma factory de DAO, a chamada DAOFactory, que implementará o padrão de projeto Factory. A classe de negócio será responsável pelo repasse de um EntityManager JPA para objetos DAO via construtor.

A DAOFactory retornará sempre uma nova instância da DAO solicitada.

As consultas serão escritas utilizando JPQL e o mapeamento objeto relacional utilizará as annotations JPA.

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="Pessoa",schema="RH")
public class Pessoa  implements Serializable {
    @Id
    @Column(name="CodigoPessoa", nullable=false)
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private Integer codigoPessoa;
    ..
}
```

Exemplo de código 3 POJO com annotations

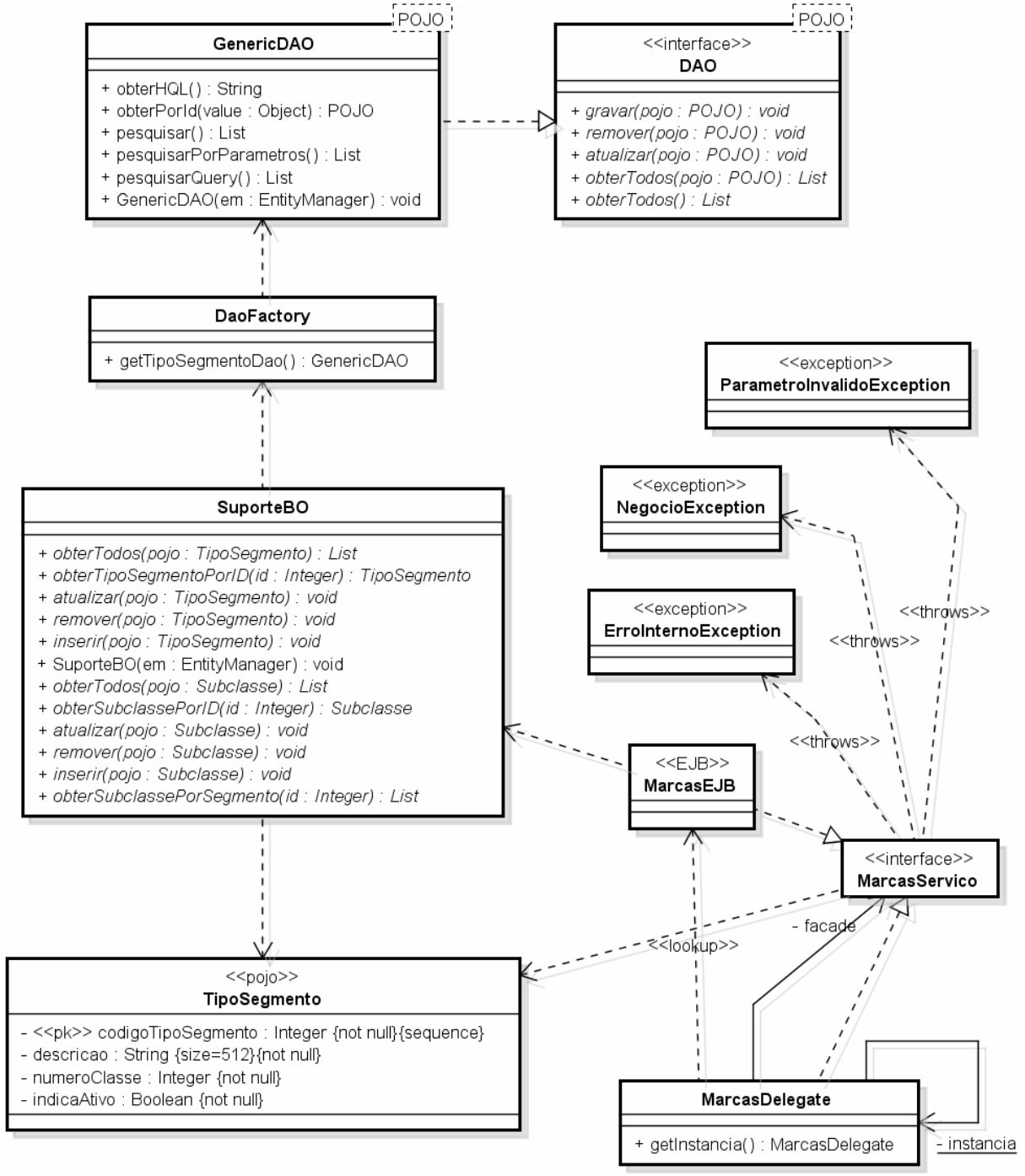

Ilustração 8 Exemplo das relações entre as classes envolvidas

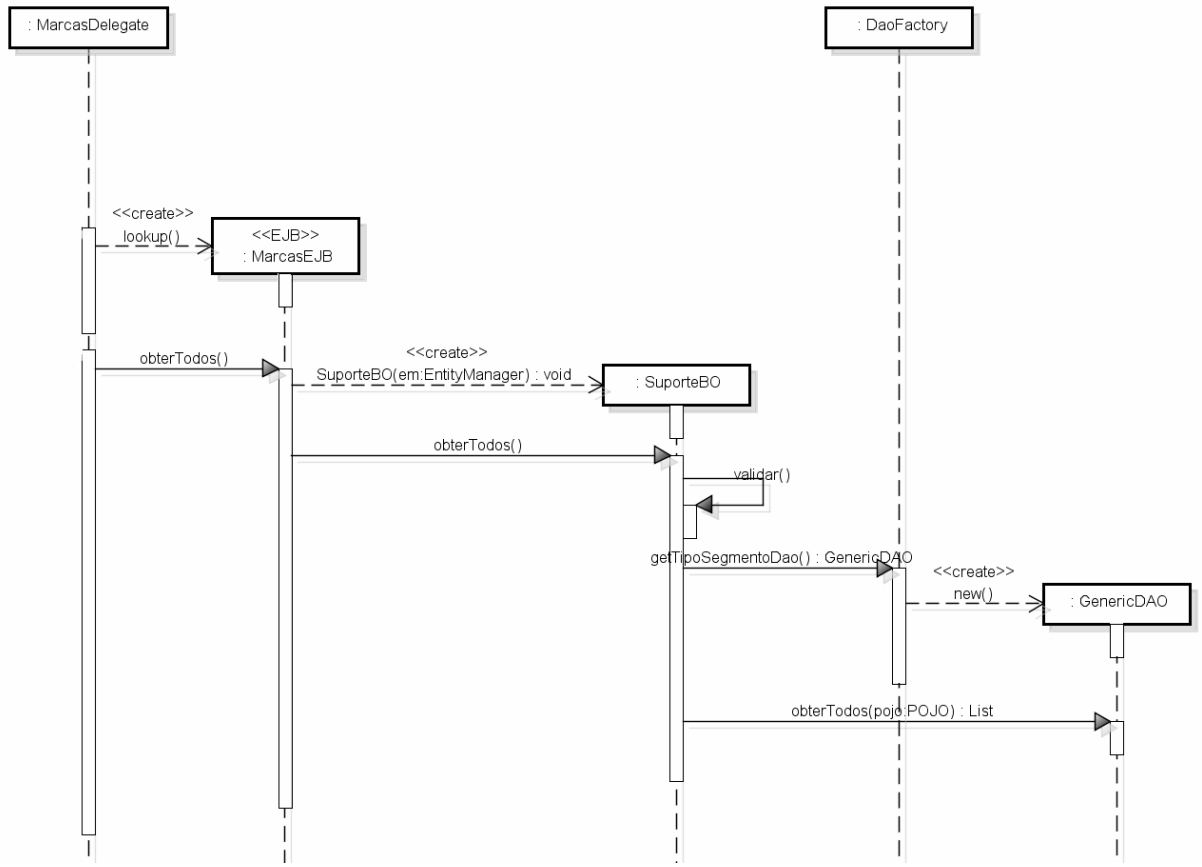

Ilustração 9 Exemplo de seqüência de funcionamento da factory de DAO para uma consulta

Toda parte transacional deverá ser repassada ao container não sendo feito nenhum tipo de tratamento no componente.

Dessa forma não será necessário a implementação de nenhum tipo de controle transacional ficando tudo a cargo do container da aplicação.

7.3.3.3 Tratamento de Exceções

O tratamento de exceções será realizado de forma programática, com isso para toda possível exceção a ser lançada existirá um tratamento já esperado. Dessa forma as exceções que podem ser lançadas pelos frameworks que compõem o componente serão capturadas e convertidas para exceções dentro do domínio do sistema.

Todas as exceções serão capturadas e convertidas para exceção *br.gov.core.dao.exception.DAOException*. Essas, por sua vez, deverão ser tratadas na implementação das classes EJB. Caso não seja o possível o tratamento, deverão lançar *br.gov.core.servico.exception.ErroInternoException*, também contida na biblioteca Core.jar.

Por se tratar de um exceção de *Runtime*, *br.gov.core.dao.exception.DAOException* não precisa ser declarada na assinatura do método, mas no momento do acesso ao método da DAO essa exceção deverá ser esperada. Essa exceção deverá estar declarada no *javadoc* do método.

Para exceções que façam parte do fluxo normal do sistema, ou seja, especificas para os fluxos de exceções nas regras negociais, serão criadas classes estendendo da classe *br.gov.core.exception. NegocioException* contida na biblioteca Core.jar.

7.3.3.4 Log

Todas as ações realizadas pelo usuário dentro do componente EJB deverão realizar log de auditoria de dados por meio do componente de log do corporativo.

7.4 Visão de processo

Esta seção define o sistema em tempo de execução, ou seja, em termos de processo ou threads que controlam o sistema.

Na arquitetura proposta à maioria dos processos são controlados por threads. Estas threads são controladas pelo próprio container do servidor de aplicações. Neste caso nenhum controle adicional é necessário.

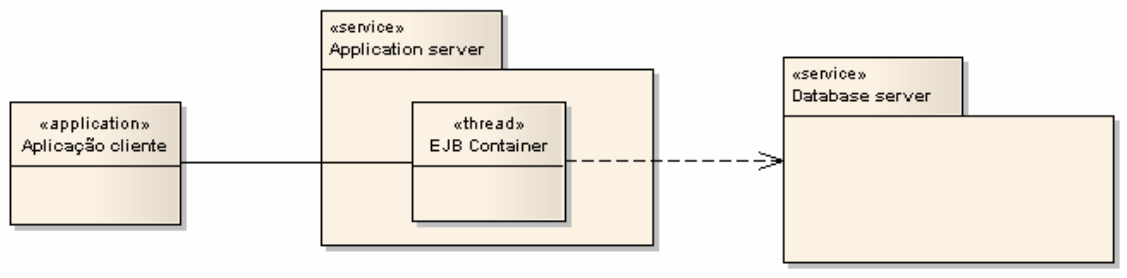

Ilustração 10: Processos

7.5 Visão de implantação

Na arquitetura proposta os componentes de negócio serão fornecidos pela tecnologia EJB 3 e componentes de persistencia construídos baseados em JPA.

Para empacotar todas essas tecnologias e facilitar a instalação / deployment da aplicação, foi definido que a aplicação será empacotada em dois conjuntos:

- `WEBDrci`.war: pacote .war com as classes da aplicação web.
- Drci.jar: pacote .jar contendo as implementações de regras de negócio e toda a lógica de acesso à base de dados relacional.

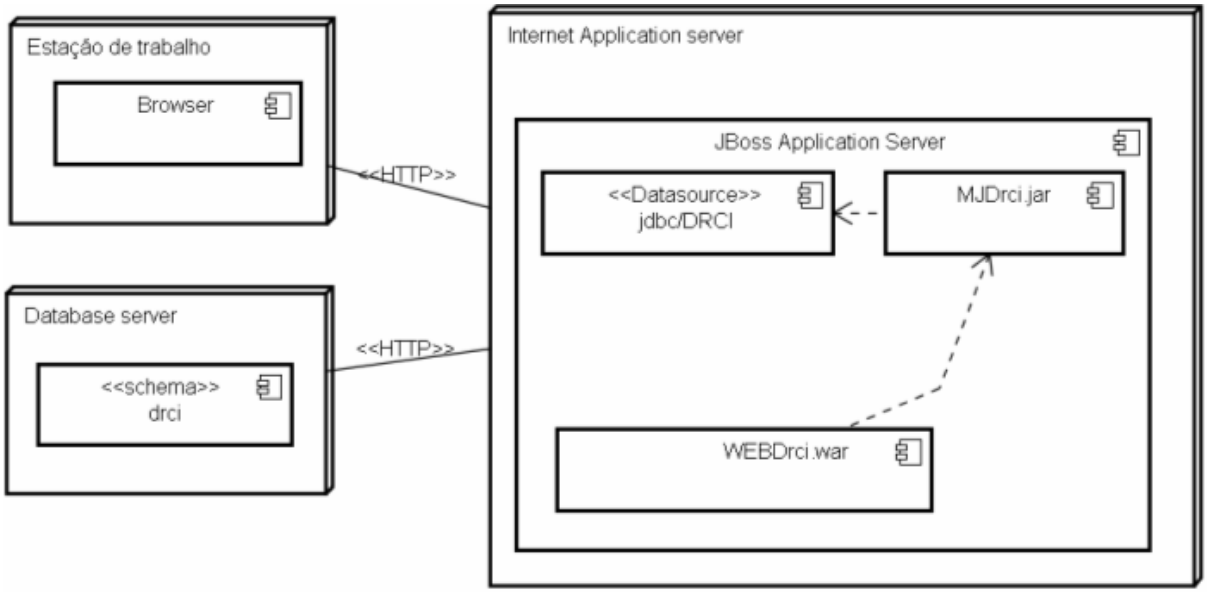

Ilustração 11 Visão de implantação

8. Visão de dados

Os mapeamentos serão controlados por pessoas responsáveis (normalmente o Arquiteto / Líder do time de desenvolvimento ou responsável pelo Banco de Dados) por manter atualizados os mapeamentos dos objetos. A alteração destes mapeamentos deve ser avisada e comunicada a todos do projeto sempre que necessário.

O dialeto do banco de dados a ser utilizado é o do MS SQL SERVER, pois o SGBD será SQL Server 2005.

A nomenclatura utilizada seguirá os padrões estabelecidos pela área de dados da CGTI.

8.1 Datasources utilizados

- jdbc/Drci. Acesso ao banco MS SQLServer.

9. Qualidade

Esta seção descreve como a arquitetura proposta atende aos atributos de qualidade considerados importantes para sistemas de informações baseados em rede e aos requisitos do gestor.

9.1 Acessibilidade e usabilidade

Para atender a necessidade de compatibilidade e acessibilidade exigidas, bem como os aspectos de usabilidade, as interfaces do usuário serão construídas com base em padrões web (web standards), recomendações do W3C e as diretivas definidas pela CGTI.

Os padrões web são constituídos por linguagens estruturais, como XHTML, XML, linguagens de apresentação CSS (Cascading Styles Sheets), modelos de objeto DOM e linguagens de script, como ECMAScript (a versão padronizada do JavaScript).

9.2 Reusabilidade

O uso de componentes já desenvolvidos objetivam minimizar o esforço de desenvolvimento em novos projetos. Componentes que já foram desenvolvidos e testados podem ser reutilizados. A arquitetura proposta foi projetada, em suas camadas, para prover a possibilidade de se construir componentes que podem ser reutilizados em algum nível.

A utilização de componentes tecnológicos, apesar de não estarem ligados a requisitos funcionais, pode proporcionar importantes ganhos de produtividade e conseqüentemente melhorar a manutenibilidade dos sistemas. Desse modo, a arquitetura proposta preconiza, sempre que possível, a utilização desse tipo de componente.

A implementação da camada de negócio será baseada em componentes (subsistemas) e possibilitará a reutilização dos serviços de negócio. Esses componentes devem ter suas interfaces documentadas e disponibilizadas para as equipes de desenvolvimento.

9.3 Manutenibilidade

A manutenção de software é o processo de modificação de um produto de software, componente ou sistema após a sua instalação, de forma a corrigi-lo, melhorá-lo ou adaptá-lo para uma mudança no ambiente operacional. A arquitetura proposta melhora a manutenibilidade nos seguintes aspectos:

Utilização de Padrões de Projetos

Padrões de projetos são soluções previamente criadas e já testadas que direcionam para soluções prontas, tais como frameworks. Isso deve facilitar a manutenção evolutiva dos sistemas, considerando-se que esses padrões são bem conhecidos por projetistas e implementadores.

Sistema em Camadas

Cada camada prevista na arquitetura possui responsabilidades bem definidas e distintas. Desse modo, tornam-se camadas com baixo acoplamento em relação a outras e altamente coesas, o que facilita manutenções corretivas e evolutivas.

A organização da camada de negócio em componentes coesos e com baixo acoplamento, melhora a manutenibilidade dos sistemas. Um componente é utilizado através de suas interfaces, não importando aos clientes do componente como seus elementos se relacionam para resolver determinado problema. Dessa forma, um controle eficiente de configuração e documentação, facilitará as atividades de manutenção corretivas e evolutivas.

10. Assinaturas