Histórico de Versões

Data	Versão	Descrição	Autor	Revisor	Aprovado Por
11/07/2008	1.0	DAS Inicial para versão 0.2 do framework.	Elizier Santos	Vanderson Botelho	N/A
13/10/2008	1.1	Atualização para versão 1.0 do framework.	Elizier, Heitor, Mário, René, Vanderson		
13/10/2008	1.2	Atualização de objetivos e restrições	Heitor		
11/11/08	1.3	Internacionalização conforme documento FRAMEWORK-DAS-EN_US.ODT	Luciana		
26/11/08	1.4	Atualização conforme implementações	René		
28/11/2008	1.5	Inclusão das classes de proxy	René		
02/12/2008	1.6	Correção ortográfica		Flávio	
13/12/08	1.7	Alteração do Nome do Framework para Demoiselle	René		
22/01/2009	1.8	Alteração da imagem da representação arquitetural par a diagrama de componentes, criado com JUDE, de modo a permitir sua fácil manutenção.	Flávio		
06/03/2009	2.0	Alteração do diagrama referente ao modelo arquitetural	Luciana		

Índice

Documento de Arquitetura de Software

1. Objetivo

O documento de Arquitetura de Software provê uma visão geral da arquitetura através de diferentes tipos de visões para descrever os diferentes aspectos do software.

2. Representação Arquitetural

A Figura 1 representa a estrutura geral do Demoiselle como Framework Arquitetural e suas dependências estruturais, sendo somente a camada "Arquitetural Framework" o foco de dissertação deste documento.

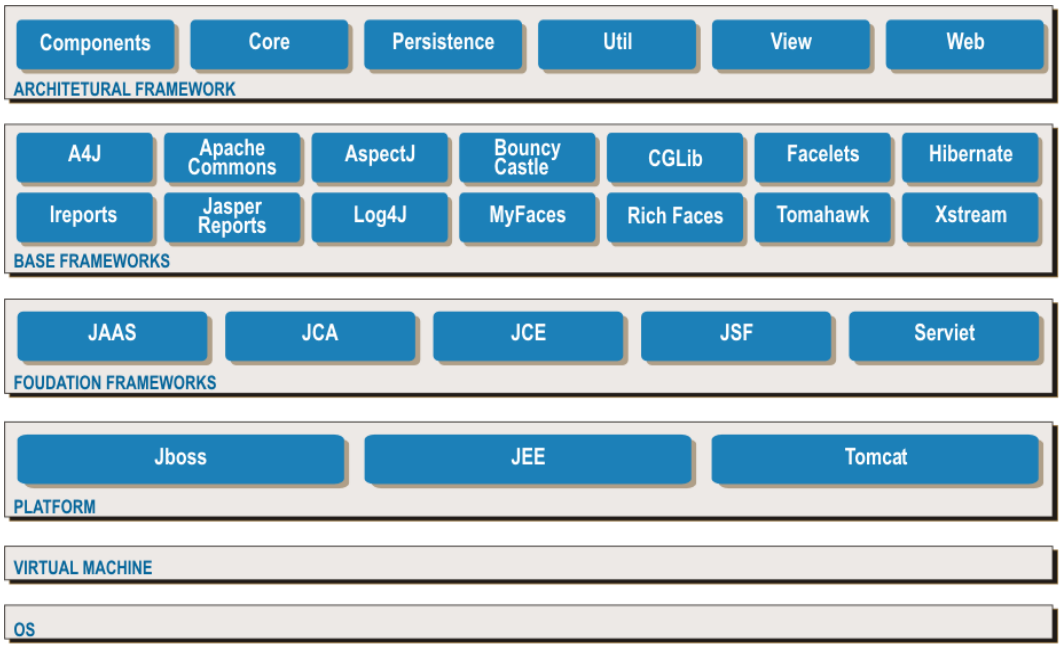

Figura 1: Modelo Arquitetural do Demoiselle

3. Objetivo e Restrições Arquiteturais

Esta seção descreverá os principais objetivos e restrições da arquitetura do Demoiselle. É sempre conveniente ressaltar que a arquitetura em questão é norteada por este conjunto aspectos.

Por se tratar de um framework arquitetural de aplicações web, este não possui

uma arquitetura completa e sim suas partes e/ou componentes possuem arquiteturas especificas. Caso uma parte ou componente possua uma arquitetura mais complexa, este fará jus à um documento DAS dedicado.

Objetivo 1	Extensibilidade
Decisão de Projeto:	A arquitetura possui pontos de extensão seja por meio de interfaces, abstrações ou pela utilização de padrões de projetos tais como Inversão de Controle e Abstract Factory, também possibilita a aplicação do padrão bridge em futuros projetos. Segue abaixo as principais interfaces da arquitetura. ● IDAO: Representa o objeto da camada de persistência, responsável pelo acesso aos dados. Classes do tipo IDAO não acessam outras camadas do framework, apenas a de persistência; ● IFacade: Representa camada de comunicação entre módulos ou sistemas; ● IBusinessController: Representa o objeto da camada de negócio e acessa a camada de persistência através de classes que implementam a interface IDAO, poderá acessar funcionalidades de outros sistemas ou módulos através de implementações da interface IFacade; ● IViewController: Representa o objeto da camada de visão que terá acesso somente as classes que implementam as interfaces IFacade e IBusinessController;

Objetivo 2	Reusabilidade
Decisão de Projeto:	A arquitetura favorece o reuso a partir da especificação de artefatos comuns em diversos projetos. Os principais artefatos de reuso são a arquitetura de referencia e os componentes do Demoiselle listados abaixo: ● Reports ● Cryptography ● Certificate

Objetivo 3	Manutenibilidade
Decisão de Projeto:	A arquitetura divide responsabilidades entre módulos lógicos para garantir o menor impacto no todo, diminuindo o acoplamento e focando a manutenção em pontos específicos.

Objetivo 4	Desempenho
Decisão de Projeto:	A arquitetura proposta minimiza os riscos de desempenho nas aplicações instanciadas por implementar os pontos críticos de performance, tais como, a Integração entre camadas e controle de transações.

Objetivo 5	Estabilidade / Confiabilidade
Decisão de Projeto:	A arquitetura é fundamentada em especificações (Foundation Framework) reconhecidas pelo mercado que garantem a estabilidade para o desenvolvimento de aplicações nela baseadas.

Restrições 1	Aplicações Web
Decisão de Projeto:	Arquitetura desenhada apenas para aplicações web não distribuídas.

Restrições 2	Ambiente de produção
Decisão de Projeto:	É necessário um ambiente de produção com servidor JBOSS 4.2.x ou TOMCAT 6.x.

4. Visão dos Casos de Uso Significativos

Funcionalidades críticas identificadas no documento de visão do projeto (DVP).

ID	Descrição da Funcionalidade
F1.01	Especificar definição de camadas
F1.02	Especificar abstração de componentes para cada camada
F2.01	Criar injeção de dependência entre componentes das camadas;
F3.01	Implementar especificação padrão para controle transacional.
F3.02	Prover controle transparente de transação às aplicações
F3.03	Prover um contexto de transação.
F4.02	Prover acesso ao contexto de segurança a partir das camadas
F4.05	Prover um contexto de segurança.
F5.01	Prover acesso a dados por meio de Hibernate, JDBC
F5.02	Prover mecanismos de busca simplificados

5. Visão Lógica

Esta seção apresenta a visão lógica do framework arquitetural. Esta visão provê uma perspectiva estrutural do framework, apresentando como este se divide em módulos lógicos e quais os principais elementos de projeto destes módulos.

5.1. Visão Geral

O Framework divide-se em módulos distintos. A seguir serão apresentados os módulos e suas respectivas responsabilidades.

5.1.1. Módulo Lógico – CORE

Este módulo contém o conjunto de especificações que dão base estrutural ao framework possibilitando padronização, extensão e integração entre as camadas das aplicações nele baseadas.

5.1.1.1. Definição de camadas

O framework propõe para as aplicações a divisão entre três camadas distintas: Visão, negócio e persistência. Uma fachada pode ser definida quando a aplicação trata

de integração entre módulos ou integração de subsistemas.

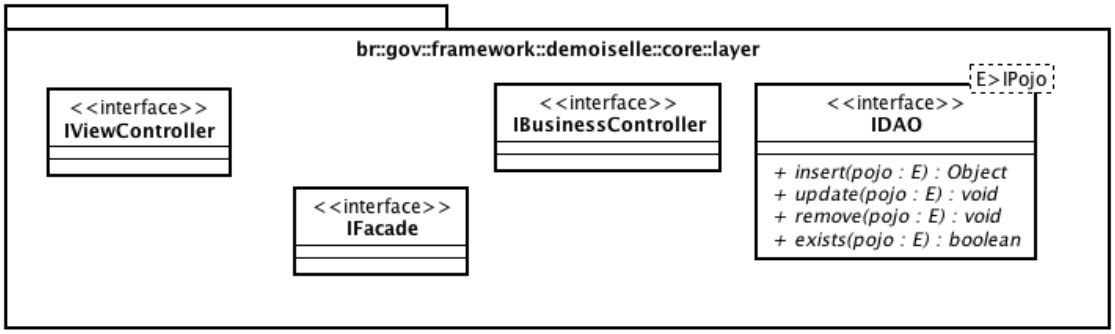

Figura 2: br.gov.framework.demoiselle.core.layer

Pacote	br.gov.framework.demoiselle.core.layer
Define abstrações para os objetos de camada.	
Interfaces	
IViewController	Abstração para o objeto da camada de visão.
IBusinessController	Abstração para o objeto da camada de negócio.
IDAO	Abstração para o objeto da camada de persistência.
IFacade	Abstração para o objeto da camada de integração de módulos/subsistemas.

5.1.1.2. Integração entre camadas

Utilização de padrões de projeto tais como Factory, Proxy, IoC e injeção de dependências para manter a integração de camadas em um nível de acoplamento baixo afim de garantir uma melhor manutenção e escrita/legibilidade das classes representantes de cada camada.

O mecanismo de integração entre camadas atuará na camada de visão injetando objetos de negócio através de uma fábrica do próprio framework ou alguma fábrica definida pela aplicação e esta fábrica poderá utilizar um proxy, do framework ou da aplicação, para a instanciação do objeto de negócio.

O mecanismo de integração entre camadas atuará também na camada de regras de negócio injetando objetos de persistência através de uma fábrica do próprio framework ou alguma fábrica definida pela aplicação e esta fábrica poderá utilizar um proxy, do framework ou da aplicação, para a instanciação do objeto de persistência.

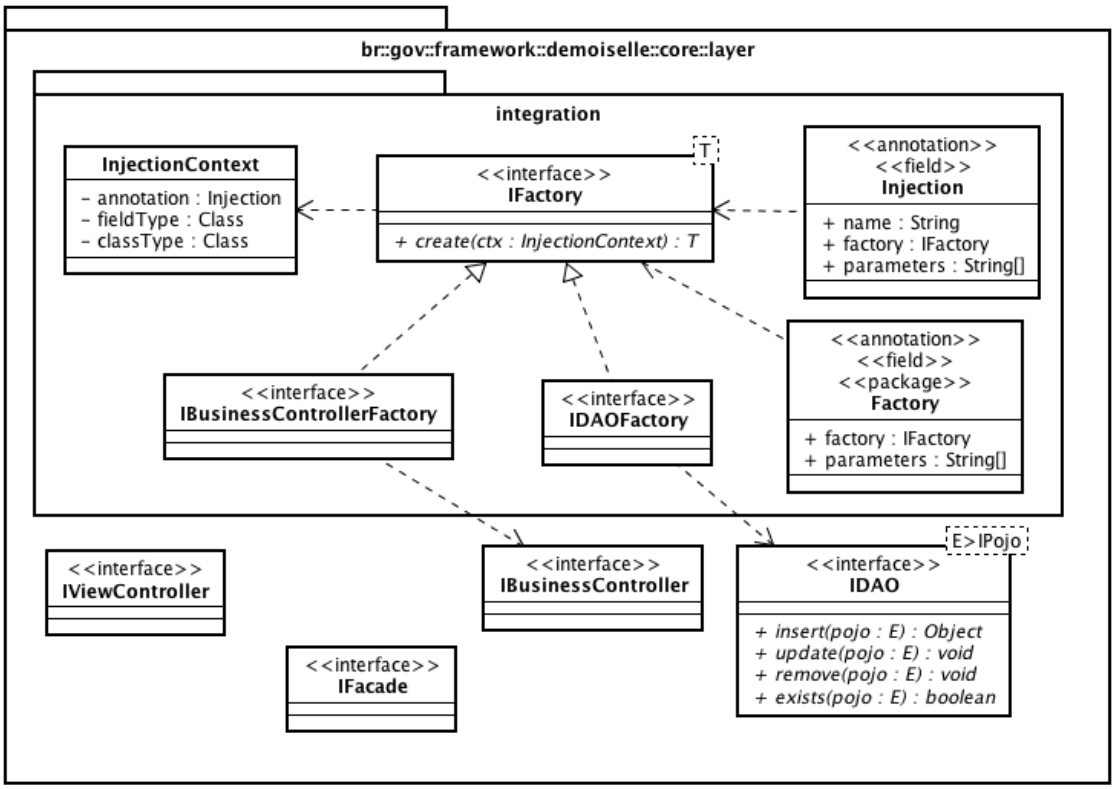

Figura 3: br.gov.framework.demoiselle.core.layer.integration

Pacote	br.gov.framework.demoiselle.core.layer.integration
Abstração para o mecanismo de integração entre camadas.	
Interfaces	
IFactory	Abstração de fábricas de objetos injetados pelo mecanismo de integração.
IBusinessControllerFactory	Especialização da fábrica de objetos de negócio.
IDAOFactory	Especialização da fábrica de objetos de persistência.
Annotations	
Injection	Annotation de propriedade que contém informações a respeito da injeção de dependência.
Factory	Annotation de classe ou pacote usada para definir a implementação da fábrica utilizada na injeção de dependência.
Classes	
InjectionContext	Informações necessárias para a fábrica realizar a criação dos objetos.

5.1.1.3. Contexto de Mensagens

Define uma abstração de mensagens trocadas durante uma requisição entre as camadas dos sistemas. Permite criar contexto de mensagens para que todas as camadas da arquitetura possam manipular mensagens. Auxilia a exibição das

mensagens para o usuário, seja na forma de tela, arquivo texto (log), em banco de dados ou na junção dessas opções. Cada implementação desta especificação deverá prover uma solução de acesso ao contexto.

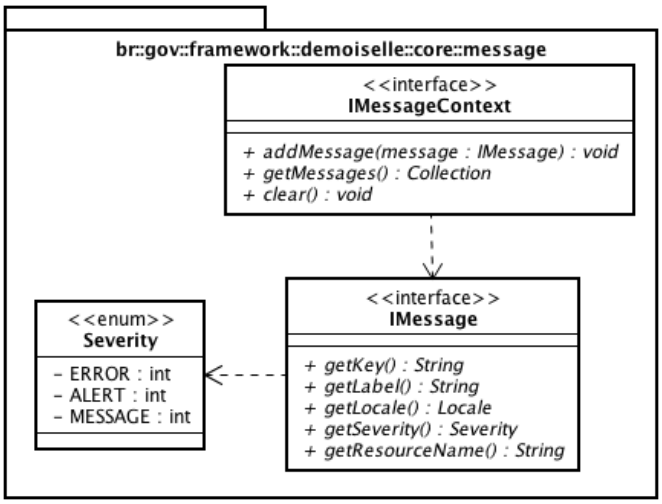

Figura 4: br.gov.framework.demoiselle.core.message

Pacote	br.gov.framework.demoiselle.core.message
Mecanismo de passagem de mensagem entre camadas.	
Interfaces	
IMessage	Abstração da unidade de mensagem.
IMessageContext	Abstração do contexto de mensagem.
Enumerations	
Severity	Lista de severidades.

5.1.1.4. Exceção

O framework define uma exceção padrão para ser utilizada pelas aplicações. Essa exceção encapsula uma mensagem padronizada para facilitar o tratamento pelos módulos do framework.

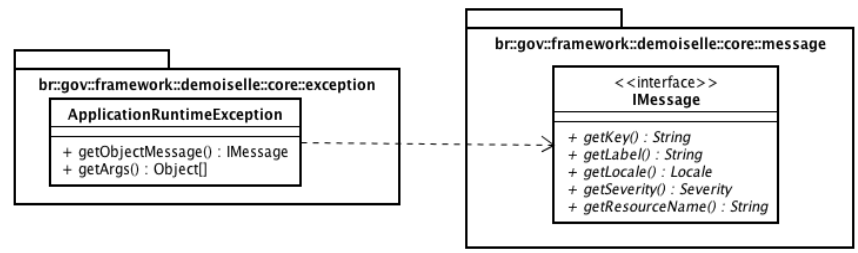

Figura 5: br.gov.framework.demoiselle.core.exception

Pacote	br.gov.framework.demoiselle.core.exception

Pacote de exceções padrão para aplicações.	
Exceções	
ApplicationRuntimeException	Exceção do tipo “unchecked” que padroniza as exceções de aplicação.

5.1.1.5. Contexto de Segurança

O framework especifica uma forma padrão de acesso aos dados de segurança referente a autenticação e autorização através de papeis. Cada implementação desta especificação deverá prover uma solução de acesso ao contexto.

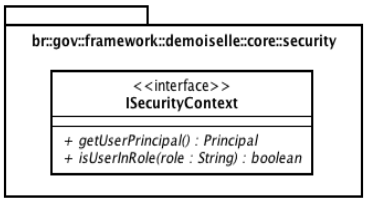

Figura 6: br.gov.framework.demoiselle.core.security

Pacote	br.gov.framework.demoiselle.core.security
Mecanismo para acesso aos dados de segurança.	
Interfaces	
ISecurityContext	Abstração do contexto de segurança.

5.1.1.6. Entidades

O framework propõe uma abstração para as entidades da aplicação.

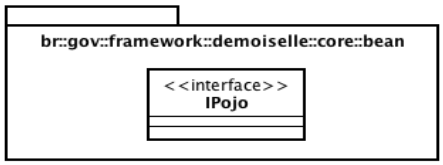

Figura 7: br.gov.framework.demoiselle.core.bean

Pacote	br.gov.framework.demoiselle.core.bean
Abstração de beans das aplicações.	
Interfaces	
IPojo	Abstração de entidades utilizadas pelas aplicações.

5.1.1.7. Transação

Especificação de mecanismo de controle transacional. Define um contexto transacional que atua no início e fim de cada ação executada pelos artefatos de cada camada. Seu funcionamento depende de um tipo definido, seja local ou JTA. O tipo

Local indica que a implementação é responsável por gerenciar a transação, e no caso do tipo JTA, a especificação necessitará de uma implementação JTA disponível.

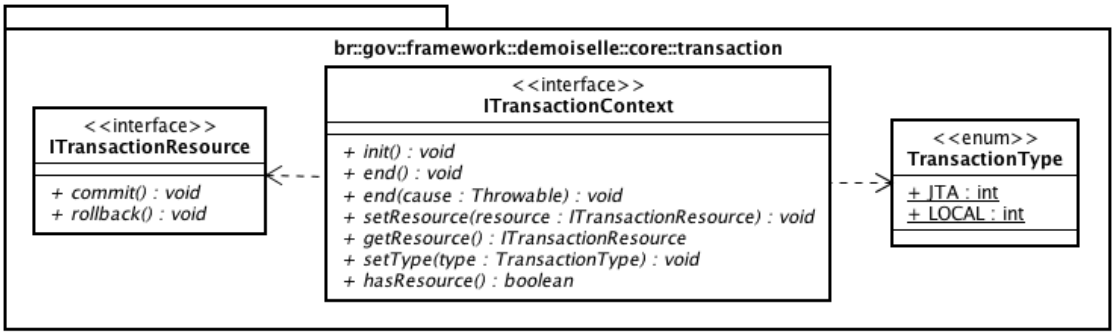

Figura 8: br.gov.framework.demoiselle.core.transaction

Pacote	br.gov.framework.demoiselle.core.transaction
Especificação de mecanismo de controle transacional.	
Interfaces	
ITransactionResource	Define um recurso a ser registrado no contexto de transação.
ITransactionContext	Contexto de transação responsável por registrar o início e fim de cada ação e registrar recursos transacionais.
Enumerations	
TransactionType	Tipos de funcionamento do mecanismo de controle de transação.

5.1.1.8. Acionadores

Define um mecanismo padronizado de ações a serem executadas aplicação. Essas ações são definidas como funções estruturais da aplicação, que não se referem diretamente à regras de negócio, como por exemplo carregamento de configuração, inicialização de ambiente, etc.

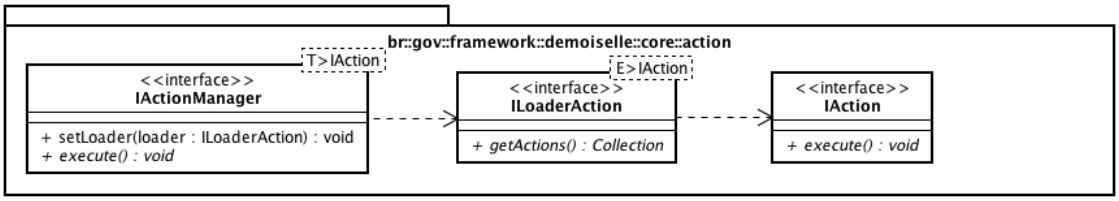

Figura 9: br.gov.framework.demoiselle.core.action

Pacote	br.gov.framework.demoiselle.core.action
Define um mecanismo padronizado de execução de ações.	
Interfaces	
IActionManager	Executa ações que foram recuperadas pelo mecanismo de recuperação.
ILoaderAction	Mecanismo de recuperação das ações.
IAction	Representa uma ação.

5.1.1.9. Localizador de Contextos

Para que as camadas das aplicações baseadas no framework possam usufruir dos contextos definidos no Módulo Lógico CORE, a existência de um localizador é fundamental. As implementações de cada contexto deverão utilizar este localizador de contextos para que outras camadas possam utilizá-lo.

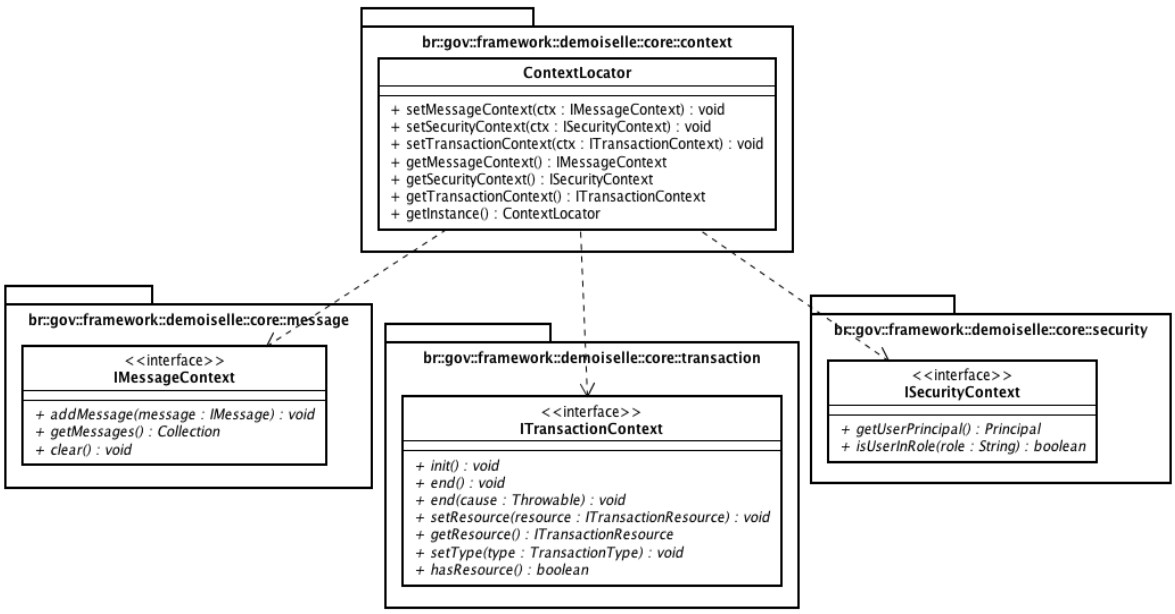

Figura 10: br.gov.framework.demoiselle.core.context

Pacote	br.gov.framework.demoiselle.core.context
Mecanismo de localização de contextos definidos no CORE.	
Classes	
ContextLocator	Implementa mecanismo de localização de contextos.

5.1.2. Módulo Lógico – UTIL

Este módulo contém componentes utilitários que facilitam o trabalho de outras funcionalidades do framework e seus módulos lógicos.

5.1.2.1. Carregamento de Configuração

O carregamento de configuração é um mecanismo padronizado que permite carregar variáveis configuradas no ambiente, arquivos xml ou arquivos de propriedades para um objeto, esse mecanismo é utilizado em vários outros componentes do framework e pode ser utilizado também pelas aplicações.

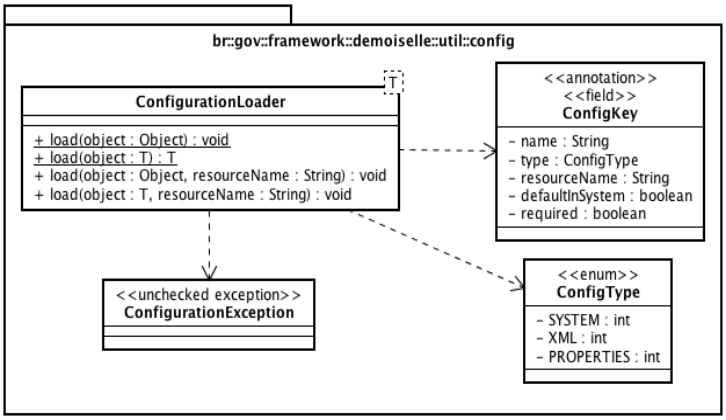

Figura 11: br.gov.framework.demoiselle.util.config

Pacote	br.gov.framework.demoiselle.util.config
Mecanismo de carregamento de configuração	
Classes	
ConfigurationLoader	Utilitário que executa o carregamento das configurações nas classes.
Annotations	
ConfigKey	Anotação usada nas classes para identificar os atributos que podem ser carregados a partir de uma configuração.
Enumerations	
ConfigType	Tipos de recursos aceito pelo mecanismo.
Exceptions	
ConfigurationException	Exceção para o utilitário.

5.1.2.2. Paginação de Resultados

Normalmente as aplicações necessitam trafegar resultados entre as camadas de forma paginada garantindo o desempenho da aplicação. Esse mecanismo é implementado no módulo UTIL com um objeto que permitem configurar os dados da página que será requisitada e um objeto que contém os resultados de forma paginada.

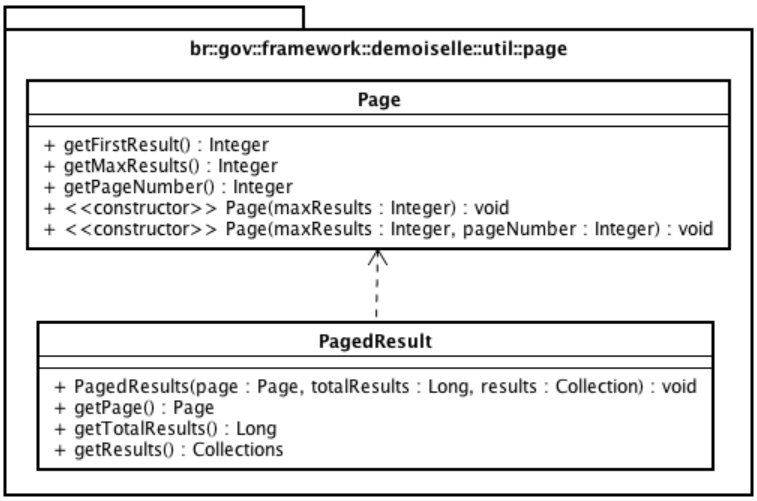

Figura 12: br.gov.framework.demoiselle.util.page

Pacote	br.gov.framework.demoiselle.util.page
Mecanismo para paginação de resultados.	
Classes	
Page	Configuração da Página.
PagedResult	Resultados paginados.

5.1.3. Módulo Lógico – WEB

Este módulo contém implementações de algumas especificações do módulo lógico CORE e também utilitários comuns de aplicações web que facilitam tratamento de sessões de usuário e suas requisições.

5.1.3.1. Contexto de Segurança – WEB

Implementa o contexto de segurança proposto no módulo CORE através de um singleton que é inicializado a cada requisição do usuário com informações de autenticação e autorização.

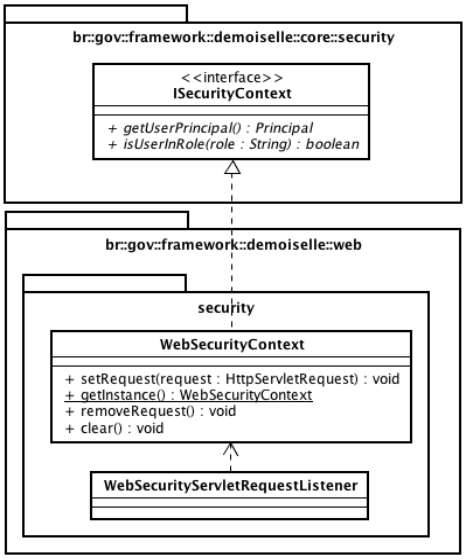

Figura 13: br.gov.framework.demoiselle.web.security

Pacote	br.gov.framework.demoiselle.web.security
Mecanismo de acesso aos dados de segurança.	
Classes	
WebSecurityContext	Implementa o contexto de segurança através do padrão singleton. Gerencia os dados de segurança vinculados a thread corrente.
WebSecurityServletRequest Listener	Responsável por repassar o objeto “request” para o contexto de segurança WebSecurityContext.

5.1.3.2. Contexto de Mensagens

Implementação do contexto de mensagens para aplicações web.

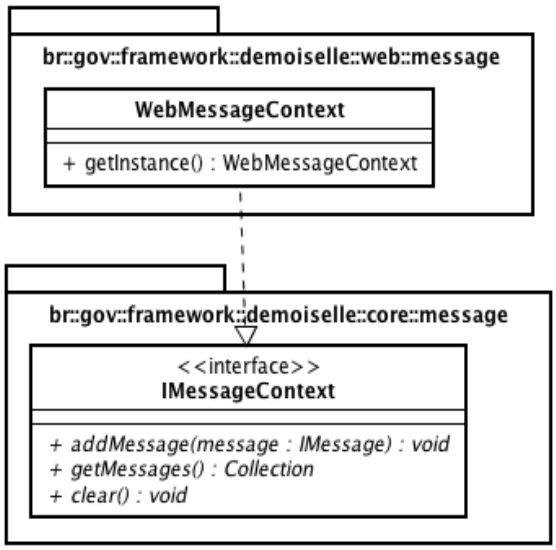

Figura 14: br.gov.framework.demoiselle.web.message

Pacote	br.gov.framework.demoiselle.web.message
Mecanismo de contexto de mensagens.	
Classes	
WebMessageContext	Contexto de mensages.

5.1.3.3. Integração entre Camadas – WEB

O módulo WEB implementa a especificação de integração de camadas proposto pelo módulo CORE. O mecanismo implementado utiliza AOP para detectar os pontos de integração. Permite que sejam implementadas novas fábricas de acordo com a necessidade da aplicação.

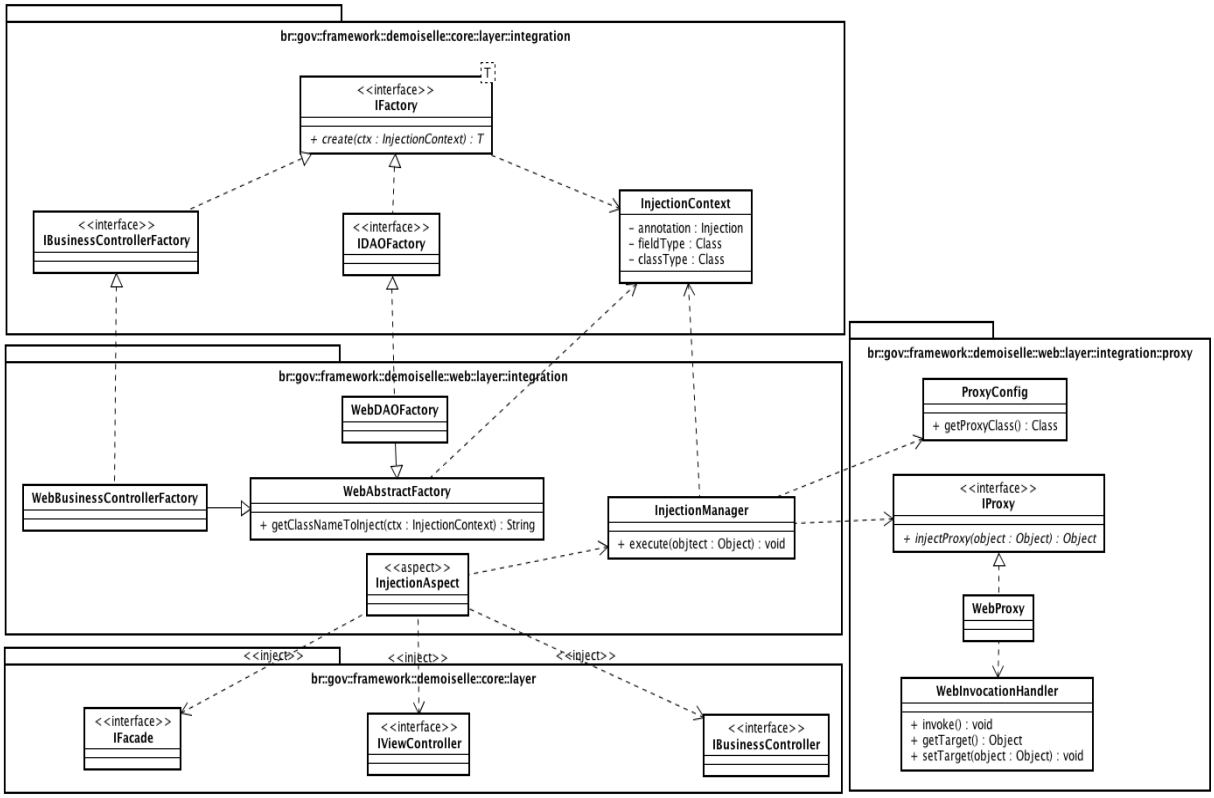

Figura 15: br.gov.framework.demoiselle.web.layer.integration

Pacote	br.gov.framework.demoiselle.web.layer.integration
Mecanismo de integração de camadas. Fabrica os objetos através de convenção de nomes entre as interfaces e sua implementação.	
Classes	
WebAbstractFactory	Abstração de uma fábrica genérica do módulo Web. Todas as outras fábricas do módulo Web deverão especializar esta classe.
WebBusinessControllerFactory	Implementação padrão da fábrica de objetos de negócio.
WebDAOFactory	Implementação padrão da fábrica de objetos de persistência.
InjectionManager	Mecanismo de injeção de dependência.
Aspectos	
InjectionAspect	Aspecto que captura os pontos onde ocorrerá a injeção.

Pacote	br.gov.framework.demoiselle.web.layer.integration.proxy
Implementação opcional para a aplicação desenvolver os objetos criados pela injeção com um proxy.	
Classes	
WebProxy	Implementação básica da interface que define um Proxy.
ProxyConfig	Classe de configuração que diz qual classe implementa a interface IProxy.

WebInvocationHandler	Controlador de chamadas de métodos utilizada pela injeção de dependência na hora de envolver o objeto criado em um proxy.
Interfaces	
IProxy	Define o comportamento padrão do proxy chamado pela injeção de dependência (InjectionManager). Sua implementação deverá ser registrada no arquivo de configuração do framework e carregada através da classe ProxyConfig.

5.1.3.4. Transação – WEB

O módulo WEB implementa a especificação do contexto transacional proposto no módulo CORE. O mecanismo implementado utiliza AOP para prover um mecanismo transparente de gerenciamento de transação. É possível utilizar o controle transacional do contêiner (JTA) para isso deve existir uma implementação de um mecanismo de lookup via JNDI.

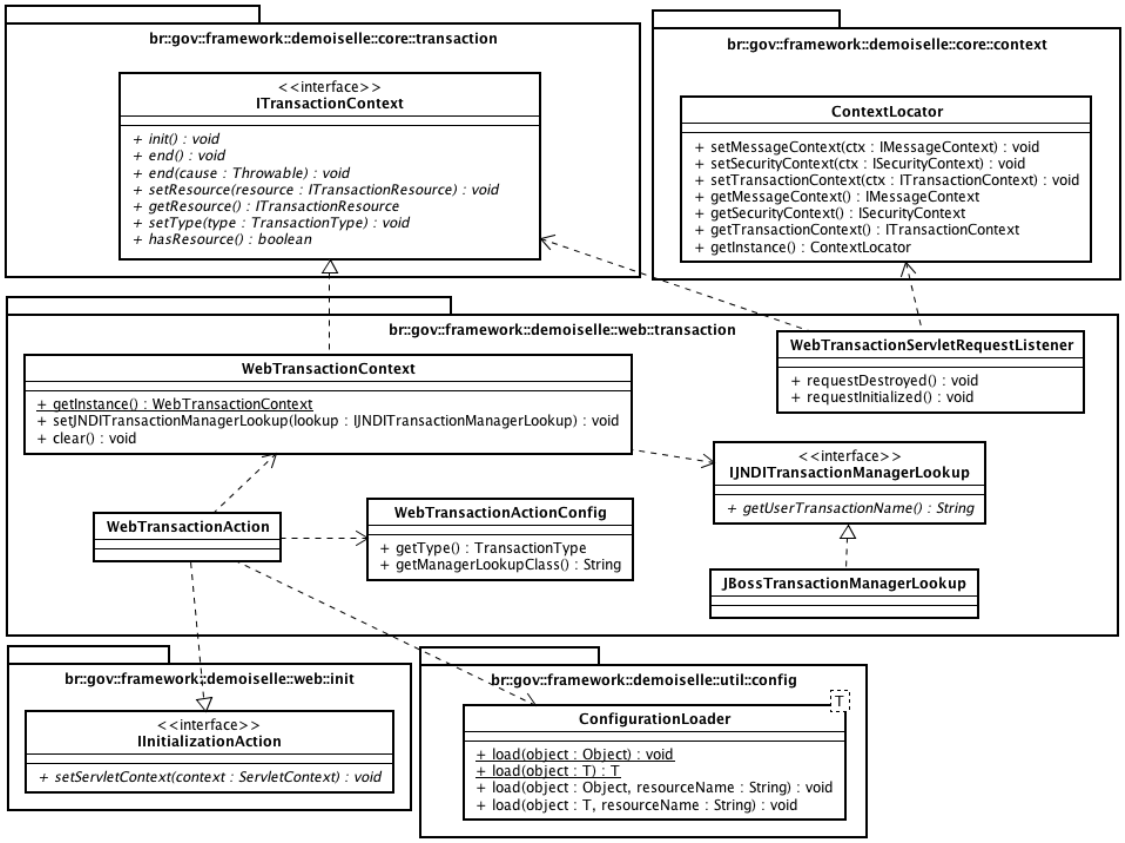

Figura 16: br.gov.framework.demoiselle.web.transaction

Pacote	br.gov.framework.demoiselle.web.transaction
Mecanismo de gerenciamento de transação.	

Interfaces	
IJNDITransactionManagerLookup	Definição de informações para o mecanismo JNDI localizar uma UserTransaction do contêiner com suporte JTA.
Classes	
WebTransactionContext	Implementação padrão do contexto transacional.
JBOSSTransactionManagerLookup	Implementação para JBOSS.
WebTransactionAction	Ação de inicialização em aplicações web onde configura o contexto transacional
WebTransactionActionConfig	Configurações padrão do contexto transacional definido pela aplicção em arquivo externo.
WebTransactionServletRequestListener	Controlador do contexto transacional. Responsável por iniciar e finalizar, normal ou com erro, o contexto transacional. É acionado em TODAS as requisições web.

5.1.3.5. Inicialização de Ambiente

A inicialização de ambiente implementa a especificação de ações proposto no módulo CORE, essa inicialização ocorre sempre que o contêiner iniciar a aplicação. O módulo WEB necessita que algumas ações sejam executadas, essas ações estão implementadas nesse módulo. Os componentes e aplicações baseadas no framework podem implementar outras ações e adicioná-las para que sejam executadas na inicialização do ambiente.

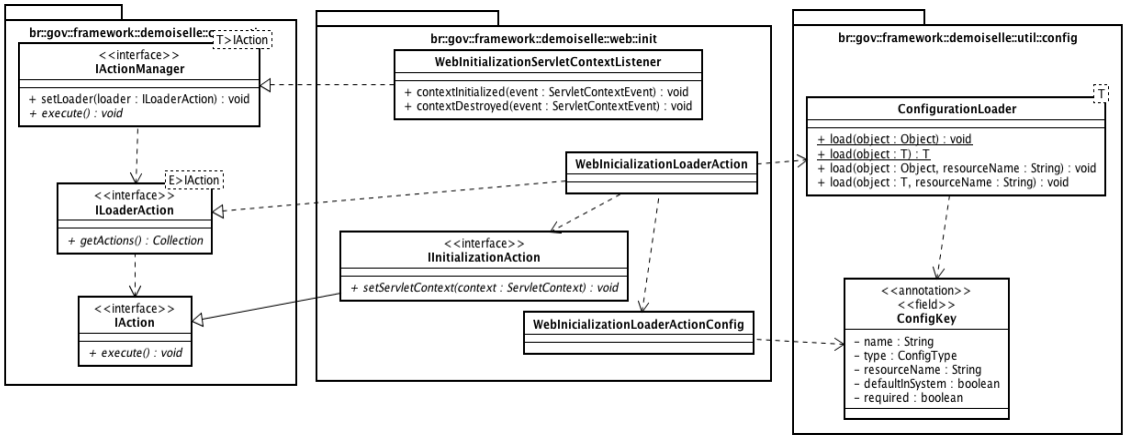

Figura 17: br.gov.framework.demoiselle.web.init

Pacote	br.gov.framework.demoiselle.web.init
Mecanismo de execução de ações na inicialização de aplicações web por meio de Servlet.	
Interfaces	

IInicializationAction	Define o comportamento de uma ação de iniciaização de aplicações web.
Classes	
WebInicializationServletContextListener	Inicializador que executará todas as actions configuradas ao inicializar o container web/servlet.
WebInicializationLoaderAction	Carrega todas as classes de ações.
WebInicializationLoaderActionConfig	Representa as configurações para o WebLoaderAction.

5.1.3.6. Redirecionamento baseado em URL

O módulo WEB implementa um mecanismo de redirecionamento baseado em URL que será usado por componentes web que necessitarem dessa funcionalidade. Esse mecanismo é baseado na especificação de ações definido no módulo CORE.

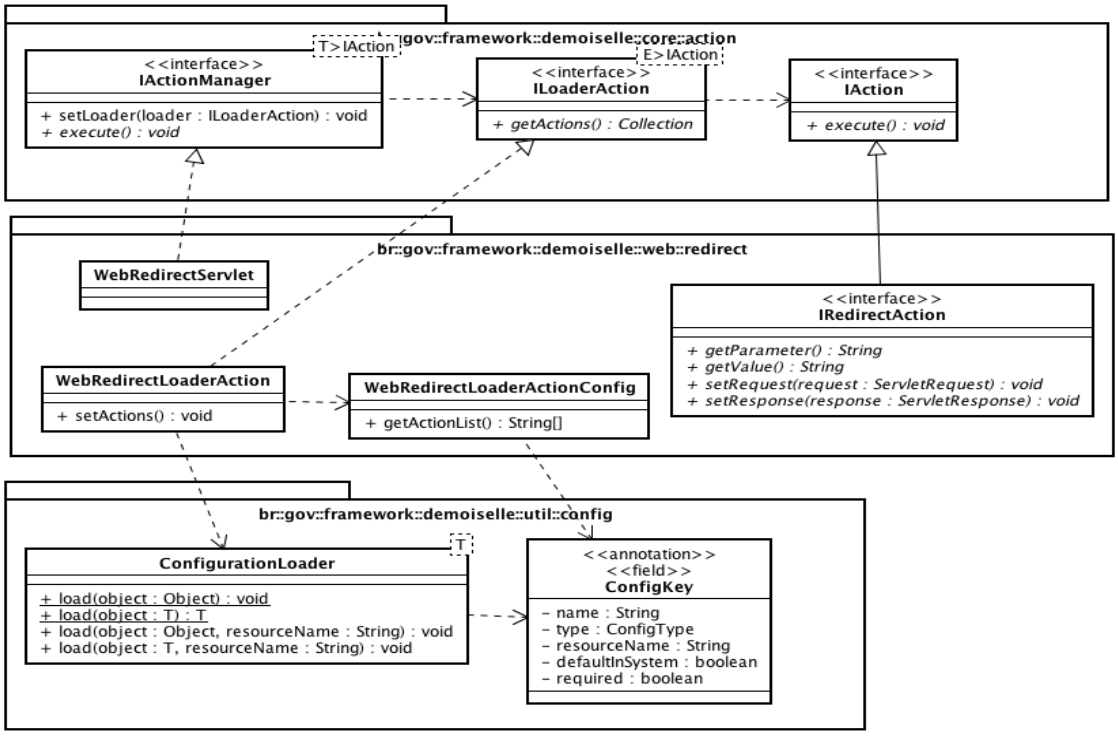

Figura 18: br.gov.framework.demoiselle.web.redirect

Pacote	br.gov.framework.demoiselle.web.redirect
Define mecanismo de redirecionamento baseado em parâmetros de URL.	
Classes	
WebRedirectServlet	Servlet que executará todas as actions de redirecionamento.
WebRedirectLoaderAction	Carrega todas as classes de ações de redirecionamento.
WebRedirectLoaderActionConfig	Configurações da classe de carregamento de ações.
Interfaces	

IRedirectAction	Especialização da interface de ação para definir informações necessárias para o redirecionamento.

5.1.4. Módulo Lógico – PERSISTENCE

Criar abstrações de persistência visando o baixo acoplamento nas camadas superiores. Facilitar a utilização de framework de base tal como hibernate e JDBC.

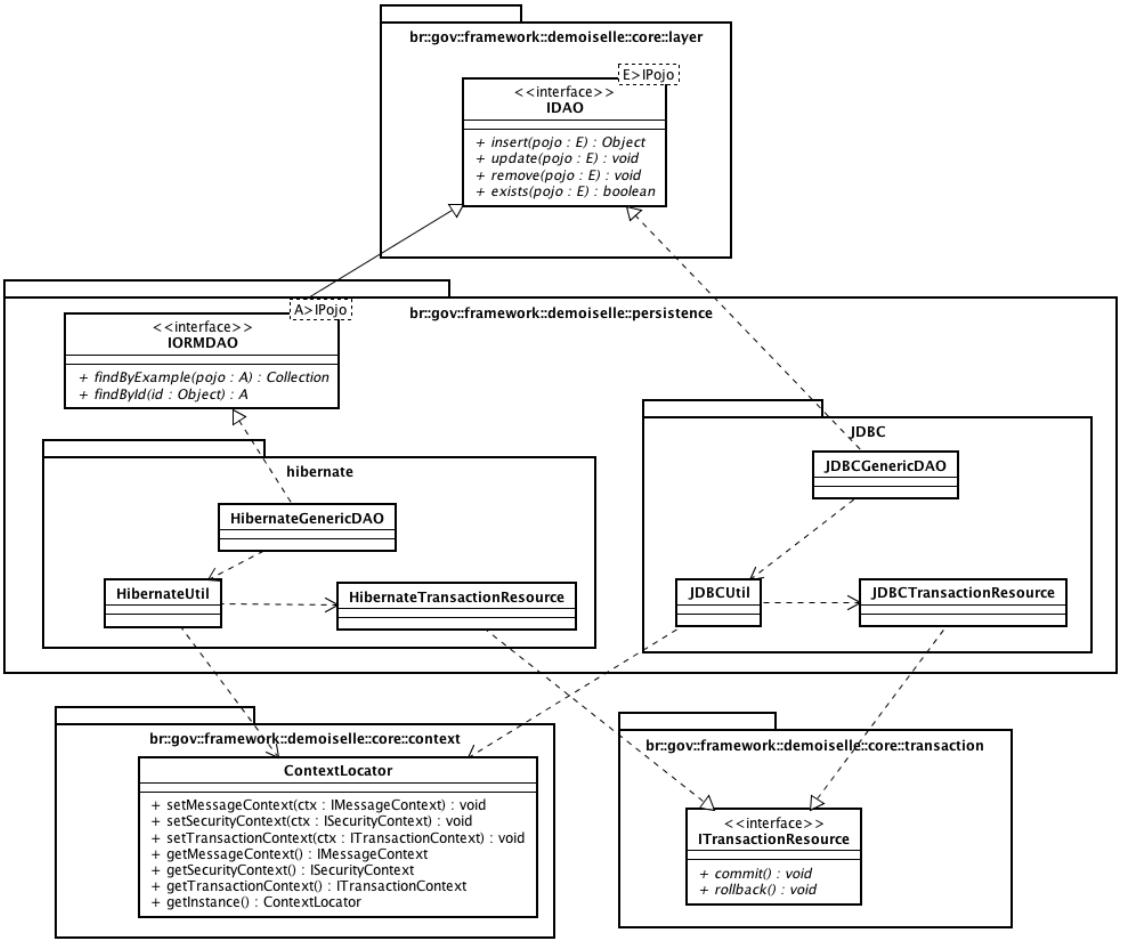

Figura 19: br.gov.framework.demoiselle.persistence

Pacote	br.gov.framework.demoiselle.persistence
Responsável pela implementação da camada de persistência definida no Módulo Lógico CORE.	
Interfaces	
IORMDAO	Interface que define os métodos a serem implementados por especializações de DAO para soluções ORM.

Pacote	br.gov.framework.demoiselle.persistence.hibernate
Pacote com as especializações para a solução ORM Hibernate.	
Classes	

HibernateGenericDAO	Implementações de métodos definidos em IDAO e IORMDAO. As aplicações que utilizarem a solução hibernate poderão herdar esta classe a fim de diminuir a quantidade de implementações das interfaces IDAO e IORMDAO.
HibernateUtil	Classe utilitária para as configurações do hibernate e também responsável por inserir a transação hibernate no Controle Transacional definido no Módulo Lógico CORE.
HibernateTransactionResource	Classe que representa uma transação hibernate.

Pacote	br.gov.framework.demoiselle.persistence.jdbc
Pacote com as especializações para desenvolvimento com drivers JDBC.	
Classes	
JDBCGenericDAO	Implementação de alguns métodos de IDAO a fim de diminuir o trabalho do desenvolvedor e padronizar o código.
JDBCUtil	Classe utilitária para as configurações JDBC e também responsável por inserir uma conexão no Controle Transacional definido no Módulo Lógico CORE.
JDBCTransactionResource	Classe que representa uma conexão JDBC.

5.1.5. Módulo Lógico – VIEW

Este módulo contém implementações de componentes específicos de interface com usuário baseados na especificação JSF.

5.1.5.1. Controlador JSF

A especificação JSF define a utilização de um controlador MVC denominado managed bean, o módulo VIEW disponibiliza uma implementação padronizada para ser utilizada nas aplicações.

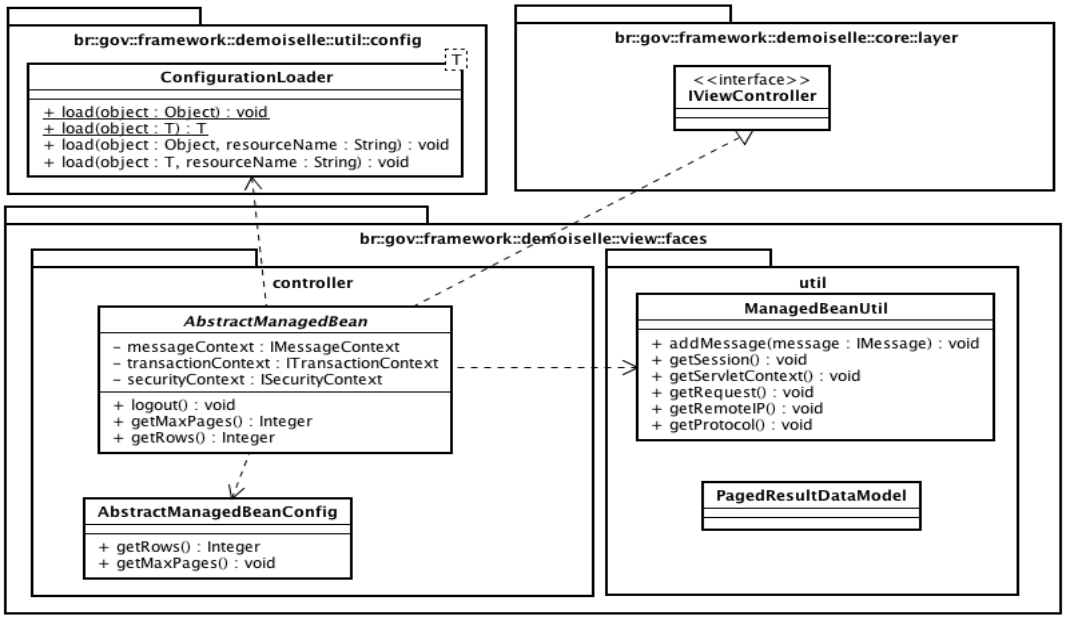

Figura 20: br.gov.framework.demoiselle.view.faces (controller & util)

Pacote	br.gov.framework.demoiselle.view.faces.controller
Controlador JSF.	
Classes	
AbstractManagedBean	Classe Abstrata que implementa o controlador da da camada de visão para JSF.
AbstractManagedBeanConfig	Representa as configurações da aplicação do total de linhas e a quantidade de páginas ao paginar dados.

Pacote	br.gov.framework.demoiselle.view.faces.util
Utilitários para JSF.	
Classes	
ManagedBeanUtil	Classe utilitária para facilitar as operações com ManagedBean.
PagedResultDataModel	Modelo de dados que converte um PagedResult para representação gráfica dos resultados paginados.

5.1.5.2. Utilitário de Cookie

O utilitário de cookie permite que sejam realizadas as operações básicas relacionas a cookies na web.

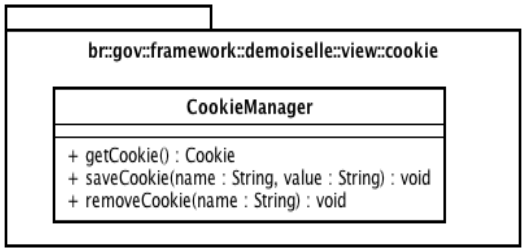

Figura 21: br.gov.framework.demoiselle.view.cookie

Pacote	br.gov.framework.demoiselle.view.cookie
Gerência de Cookies.	
Classes	
CookieManager	Gerenciador de Cookies.

6. Visão de Processos Concorrentes

Não se aplica.

7. Visão de Implantação

Os componentes do framework podem ser utilizados pela aplicação de duas formas, a primeira é através do processo manual, onde os artefatos são copiados diretamente para o projeto da aplicação, nesse caso o responsável pela aplicação deve estar atento ao controle de versão de cada componente e também deve copiar os frameworks de base utilizados por cada componente, ou seja, deve garantir que todas a dependências entre o framework e frameworks de base sejam atendidas. A figura abaixo ilustra essa relação. O processo manual é trabalhoso e sujeito a falhas pois exige que o desenvolvedor esteja atento a qualquer mudança com relação as dependências entre os componentes do framework e frameworks de base.

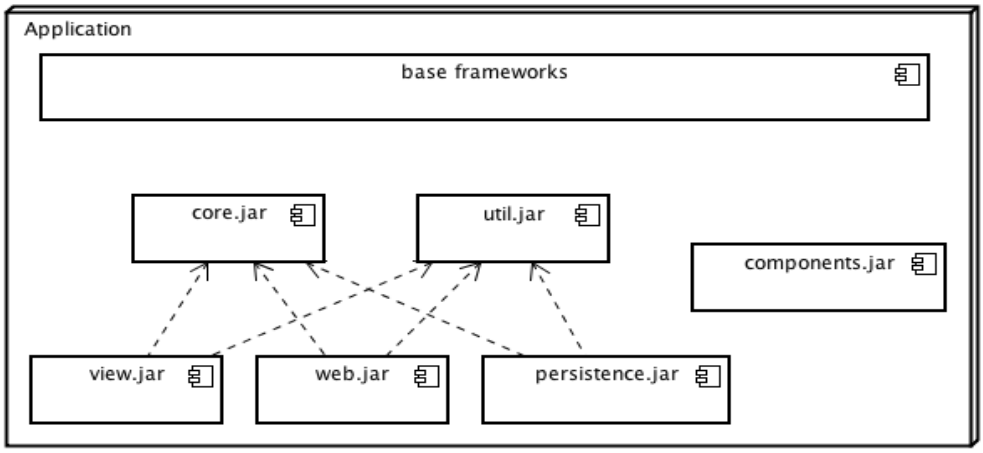

Figura 22: Visão de Implantação (Manual)

Na segunda forma o desenvolvedor estabelece as dependências entre o framework e a aplicação através do Maven, nesse caso essa ferramenta auxilia o processo de controle de dependência de forma automatizada, facilitando o trabalho e diminuindo a possibilidade de falhas.

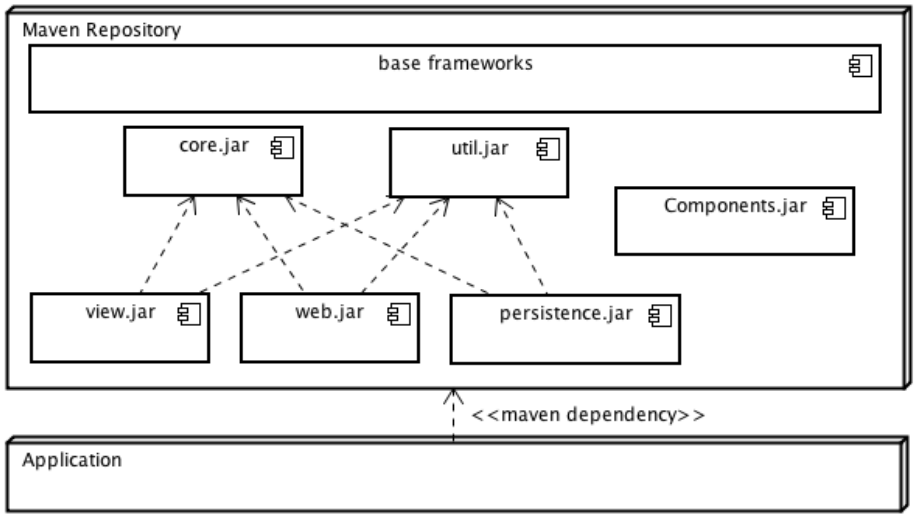

Figura 23: Visão de Implantação (Maven)

8. Visão de Implementação

O framework está dividido em dois módulos principais, o primeiro é o framework arquitetural (core, util, web, persistence e view) e o segundo é um conjunto de componentes. O framework arquitetural promove a padronização na construção das aplicações. Os componentes são complementares ao framework e possuem ciclo de vida próprio, desta forma podem ser utilizados individualmente de acordo com a

necessidade da aplicação. Novos componentes podem ser adicionados a cada release.

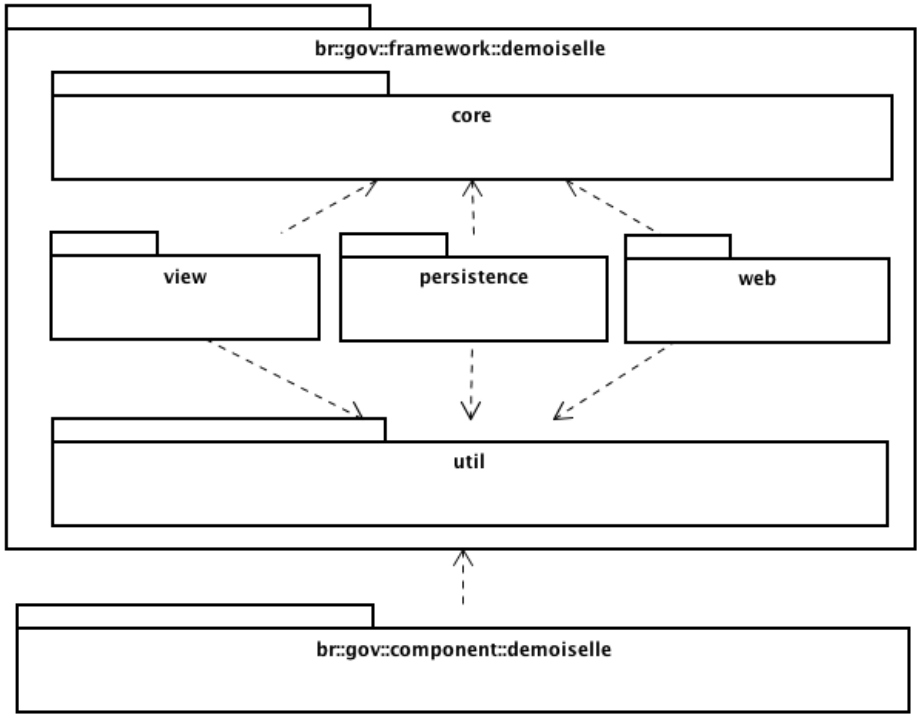

Figura 24: Visão de Implementação

9. Visão de Dados

Não se aplica.

10. Tamanho e Performance

Indefinido.

11. Qualidade

Indefinido.

Referências

Nenhuma.