Histórico de Versões

Data	Versão	Descrição	Autor	Revisor	Aprovado por
			<ARQ>	<DES>	<LP>

Índice

Documento de Arquitetura de Software

1.Objetivo do Documento

Este documento é uma referência para os documentos de arquitetura de software (DAS) de aplicações web que utilizam o Demoiselle. Derivações desta arquitetura devem ser feitas sob demanda, de acordo com as necessidades do projeto, pelo arquiteto.

2.Objetivos e Restrições da Arquitetura

Esta seção descreverá os principais objetivos e restrições da arquitetura da aplicação de referência utilizando o Demoiselle 2.

Objetivos
Simplicidade e Padronização de Aplicações Web pelo uso do framework
Facilitar o uso das funcionalidades do JavaEE 6

Restrições 1	***Aplicações Web***
Decisão de Projeto:	Arquitetura desenhada para ser implantada de forma não distribuída.

Restrições 2	***Ambiente de produção***
Decisão de Projeto:	É necessário um ambiente de produção com um servidor de aplicação compatível com JEE6.

*****OBS:As restrições devem ser mais detalhadas. É necessário que sejam explicitadas as versões de todas as APIs utilizadas no Demoiselle 2, bem como um maior detalhamento das restrições de ambiente. É importante que sejam apontados quais servidores são usados por padrão com o Demoiselle 2, e em quais versões, por exemplo.*

*É importante esclarecer nesse capítulo também as restrições sobre o uso de EJB e mensageria no Demoiselle 2.*****

<Descrever demais objetivos e restrições para a definição da arquitetura, apresentando os principais aspectos da aplicação. Quando for pertinente, listar os Casos de Uso mais críticos para a arquitetura juntamente com os respectivos desafios.>

Documentação:<Insira aqui as documentações e as imagens dos diagramas de caso de uso contidos na pasta *Elementos do Modelo Arquiteturalmente Significativos,* dentro do *Modelo de Requisitos,* que se encontra dentro de *Use Case View* >

3.Elementos Arquiteturalmente Significativos

O sistema deverá ser desenvolvido utilizando o framework Demoiselle 2. O Demoiselle é um framework que tem o objetivo de integrar frameworks especialistas. Ele facilita o desenvolvimento de aplicações, minimizando o tempo gasto e aumentando a produtividade.

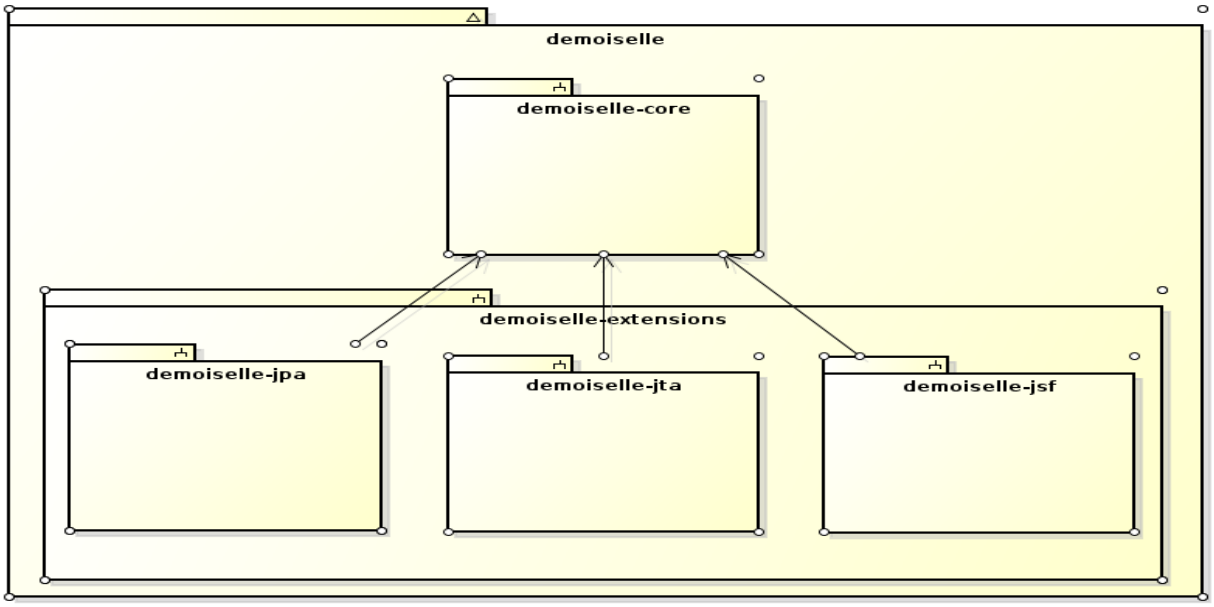


Figura 1 – Estruturação dos projetos no Demoiselle 2

Visando uma melhor modularização, o Demoiselle 2 está dividido por funcionalidades. Isto significa que o framework não é monolítico, no qual todas as suas funcionalidades estão contidas em um único pacote. Ele está dividido em Core e Extensões.

O Core do Demoiselle contém aquelas funcionalidades comuns a todas as extensões e aplicações. O core é simples, leve e formado majoritariamente por interfaces e poucas implementações. O Core é a base do framework. Sem ele, as extensões e a própria aplicação não funcionariam.

As Extensões, como o próprio nome sugere, estendem o Core com funcionalidades extras e bem específicas a um domínio ou tecnologia que seja especificação Java.

O sistema em questão utilizará o demoiselle-core, e , caso utilize alguma extensão, estas serão especificadas abaixo.

3.1.Componentes do Demoiselle 2

<Informar quais componentes do Demoiselle 2 serão utilizados.>

3.2.Profiles do Demoiselle 2

<Informar quais profiles do Demoiselle 2 serão utilizados.>

3.3.Extensões do Demoiselle 2

<**Caso** alguma extensão seja utilizada, informar qual ou quais.>

3.4.Informações complementares

<Descrever outras principais interfaces e componentes da solução (**caso existam**), registrando também a presença ou ausência de mecanismos arquiteturais inovadores (arquitetura crítica). Quando necessário, indicar necessidade de prototipação e/ou gestão de riscos arquiteturais.>

4.Descrição da Arquitetura

<Descrever as camadas, mecanismos e outras abstrações da arquitetura, além das que estão descritas do Demoiselle 2, caso sejam necessárias.>

4.1.Camadas e Subsistemas

<Apresentar as camadas lógicas da aplicação e os subsistemas ou módulos definidos, além do foram descritos do Demoiselle 2, caso seja necessário.>

Documentação:<Insira aqui as documentações e as imagens dos diagramas de classe contidos nas pastas *Camadas, Subsistemas e Pacotes*. Essas pastas se encontram situadas debaixo das pastas *04 - Elementos do Modelo Arquiteturalmente Significativos*, dentro do *Modelo de Projeto* e/ou do *Modelo de Análise*>

4.1.1. Camada de visão e Controle

Responsável por apresentar informações e controlar a interação com os atores externos ao sistema. É responsabilidade da extensão JSF do Demoiselle 2 prover mecanismos que auxiliem a utilização do JSF 2 como camada de visão. Serão criadas páginas HTML e Managed Beans. Os Managed Beans têm a função de orquestrar as chamadas aos métodos dos Business Controllers, bem como fazer a validação de campos e o gerenciamento das regras de navegação. Não deve existir regra de negócio dentro dos Managed Beans.

4.1.2. Camada de negócio

Implementa as regras de negócio definidas para o sistema. Pode definir processos de integração com outros módulos e/ou subsistemas. É responsabilidade da camada de negócio fazer validações baseadas em regras de negócio, bem como fazer a ligação entre a camada de visão e controle e a camada de negócio.

4.1.3. Camada de persistência

Realiza a integração do sistema com sistemas gerenciadores de dados para garantir o armazenamento dos estados dos objetos (POJO).

4.2.Padrões e Mecanismos Arquiteturais

Transação
A estratégia de transação utilizada pelo sistema será JTA.

Regras de acesso
As classes de uma camada inferior não poderão acessar classes de uma camada superior. Ex: Uma classe da camada de persistência não poderá acessar classes da camada de negócio assim como uma classe da camada de negócio não poderá acessar classes da camada de visão e controle.

Controladores
Os Managed Beans da camada de visão serão anotados com @ViewController
As classes de negócio serão anotadas com @BusinessController
As classes de DAO serão anotadas com @PersistenceController

Pacotes
Os ManagedBeans deverão estar no pacote view.managedbean
As classes de conversores deverão estar no pacote view.converter
As classes que contém os validadores deverão estar no pacote view.validator
As classes da camada de negócio deverão estar no pacote business
As classes da camada de persistencia deverão estar no pacote persistence
As classes de entidade deverão estar no pacote entity
As classes de exceção deverão estar no pacote exception ou no pacote da funcionalidade referente à exceção.
As classes de mensagem deverão estar no pacote message.
As classes utilitárias deverão estar no pacote util
As classes de configuração deverão estar no pacote config

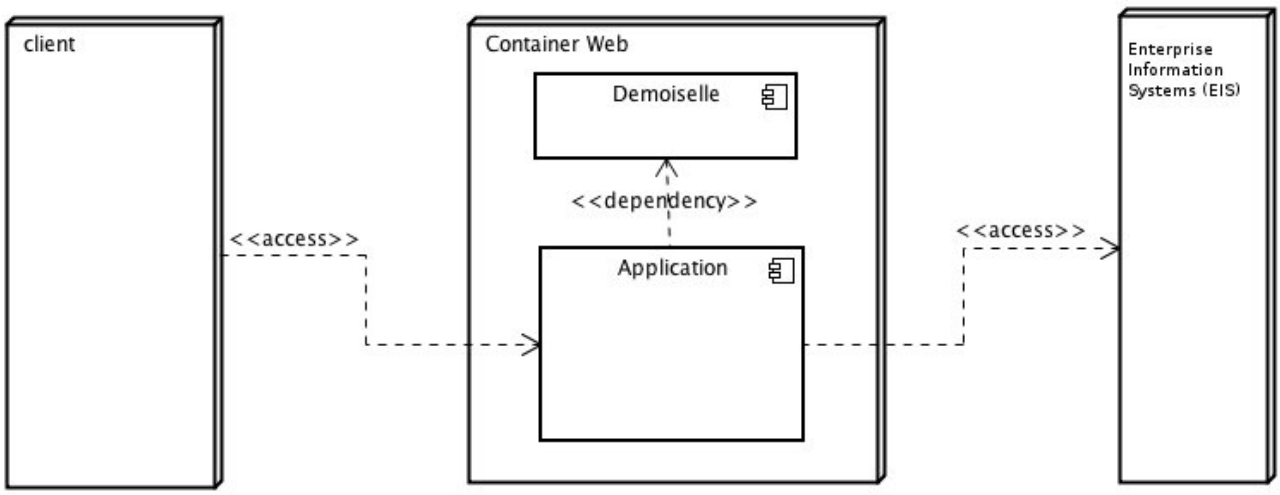
<Apresentar os mecanismos de análise e projeto, regras de modelagem, estereótipos e padrões comportamentais da arquitetura, além dos descritos do Demoiselle 2, caso seja necessário.>

Documentação:<Insira aqui as documentações e as imagens dos diagramas de interação contidos nas pastas *Padrões Comportamentais Importantes*. Essas pastas se encontram situadas debaixo das pastas *04 - Elementos do Modelo Arquiteturalmente Significativos*, dentro do *Modelo de Projeto* e/ou do *Modelo de Análise*>

4.3.Topologia

<Alterar o diagrama geral de implantação abaixo, caso se aplique ao sistema e/ou adicionar mais diagramas pertinentes.>

A figura abaixo descreve a configuração de distribuição dos componentes do software mais comum para as aplicações que utilizam o framework.



4.4. Visão de Implementação

<Esta seção é opcional e descreve os principais componentes do software a serem implantados e suas relações de dependência.>

Documentação:<Insira aqui a documentação e a imagem do diagrama de componentes>

4.5.Outras Visões

<Apresenta outras abstrações da arquitetura que se julguem necessárias, como visão de processos concorrentes, visão de dados, etc>

5.Decisões e Justificativas

Para a implementação do projeto utilizaremos o framework Demoiselle 2, por se tratar de um produto concebido pela própria empresa que visa aumentar a produtividade e padronizar o desenvolvimento.

*****OBS: Essa justificativa poderia ser melhor elaborada. Deve-se apontar mais pontos positivos do Demoiselle 2, caso existam.*****

<Adicionar aqui as demais informações pertinentes sobre as principais decisões tomadas para a arquitetura e as justificativas para as escolhas dentre as alternativas consideradas.>

5.1.Alterações na arquitetura do Demoiselle 2

<A decisão pelo uso de versões das APIs, de servidores e de quaisquer elementos fora dos padrões estabelecidos pelo Demoiselle 2, deve ser **obrigatoriamente** justificada e documentada nessa seção. Se nenhuma alteração for feita, essa seção pode ser retirada.>