PLATAFORMA DE DESENVOLVIMENTO PINHÃO PARANÁ

DESCRIÇÃO DA ARQUITETURA

AGOSTO – 2006

Sumário de Informações do Documento

Tipo do Documento: Definição		
Título do Documento: Descrição da Arquitetura		
Estado do Documento: EB (Elaboração)		
Responsáveis: Robson Valentin		
Palavras- Chaves:		
Resumo:		
Número de páginas:		
Software utilizados:		
Versão	**Data**	**Mudanças**
1.0	20/12/2004	
1.1	21/03/2006	Alteração no nome do Sistema de Segurança para Sentinela, atualização da tabela de ferramentas e inclusão do analista responsável nos Processos de Ambiente – Natasha Fortes
1.2	22/03/2006	Adicionado o tópico 3.7 – revisado por Elisabeth Hoffmann
1.3	17/07/2006	Atualização do documento por Cleverson Budel e Danielle Mayer

SUMÁRIO

1 INTRODUÇÃO

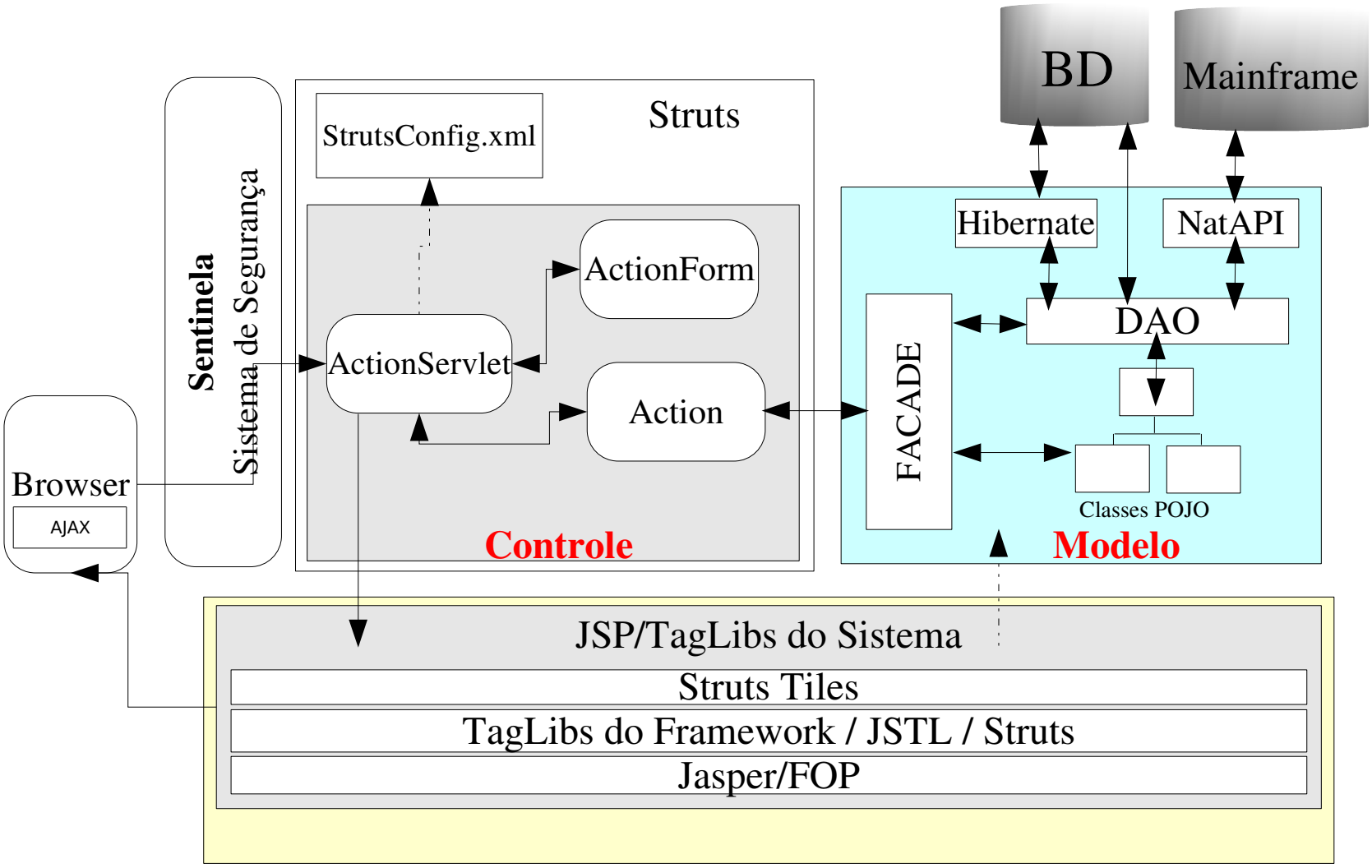
Este documento tem por objetivo padronizar e definir a arquitetura dos sistemas desenvolvidos pela empresa.

2 APLICAÇÃO

A arquitetura deverá ser utilizada nos sistemas web, baseados na plataforma Java(JEE), podendo ser adaptada para atender sistemas que não serão executados na plataforma web.

3 ARQUITETURA DE DESENVOLVIMENTO

A arquitetura definida está baseada no padrão MVC (Model, View, Controller) e na Orientação a Serviços (SOA). A figura a seguir mostra a estruturação da arquitetura:


BD
Mainframe
Struts
StrutsConfig.xml
Sentinela
Sistema de Segurança
ActionForm
ActionServlet
Action
Browser
AJAX
Controle
Hibernate
NatAPI
DAO
FACADE
Classes POJO
Modelo
JSP/TagLibs do Sistema
Struts Tiles
TagLibs do Framework / JSTL / Struts
Jasper/FOP

3.1 Camada de Controle

Representa a camada responsável pela forma como a aplicação reage aos estímulos do usuário, determina o fluxo de apresentação servindo de camada intermediária entre as camadas de visualização e modelo.

Para auxiliar nesta camada é utilizado o framework struts. A camada de controle se comunica com a camada de modelo através de classes do tipo FACADE. Esta camada também é responsável pela comunicação com o Sistema de Segurança.

3.2 Camada de Modelo

Representa a camada que contém os processos de negócio da aplicação, encapsulando dados e comportamento independente da apresentação.

Nesta camada a arquitetura utiliza a **"Facade"**, que deve conter métodos que encapsulem o acesso aos dados e que representem os serviços necessários aos casos de uso.

Para a persistência a arquitetura utiliza classes **"DAOs"** com o framework "Hibernate". Para acesso aos dados do mainframe a arquitetura utiliza a biblioteca "NATAPI" acionada através de classes "DAOs".

É utilizado uma variação da Facade, ao invés da Facade chamar método de negócios que chamariam DAOs, ela mesma trata de chamar os DAOs e executar a lógica de negócio, eliminando assim mais um nível de abstração.

3.2.1 Conceitos de Padrões de Projeto (Design Patterns)

- **Façade** – objetiva minimizar as dependências entre camadas, fornecendo uma interface de alto nível para executar a regra de negócios de uma aplicação.
- As Façades, quando utilizadas juntamente com o framework Struts, são chamadas pelas classes do tipo Action, e assim, integram a camada de controle com a camada de modelo.

- As Façades comunicam-se com a camada de persistência, através das classes DAO, para executar as regras de negócio da aplicação.
- O(s) serviço(s) genérico(s) devem ser encapsulado(s) em uma super-classe abstrata do tipo Façade (SuperFacade). Todas as outras Façades devem herdar (extends) desta super-classe.

- **DAO** – Objetiva encapsular a lógica de acesso a dados, isolando esta camada da camada de negócios. Assim, sendo necessária a alteração na forma de acesso a dados, não será preciso modificar todo sistema mas somente as classes do tipo DAO. As classes DAO realizam consultas, exclusões, inserções e acessos aos métodos do Hibernate ou a Natapi, para comunicação com o *mainframe*. A proposta é que os DAOs não possuam nenhuma lógica de negócio, somente encapsulem o acesso aos dados. Na maioria dos casos cada classe POJO deve possuir uma DAO para persisti-la.
- **POJO –** Plain Old Java Object – Objetos que encapsulam os dados da aplicação.
- **DTO –** Data Transfer Object – Objetos utilizados para guardar dados de diferentes pontos da aplicação, e trafegá-los entre as camadas da arquitetura de desenvolvimento.
- Devem ser utilizados quando houver necessidade de reunir num único objeto, atributos que estão espalhados em diversos objetos. Útil para substituir a passagem de muitos parâmetros e no retorno de métodos façades, porém quando todos os atributos necessários estiverem num único objeto, não precisa criar um DTO, neste caso é mais coerente utilizar o próprio objeto POJO.

3.3 Camada de Visualização

Representa basicamente a interface com o usuário. É utilizada para receber a entrada dos dados e apresentar o resultado.

Nesta camada deve-se utilizar páginas construídas com a tecnologia“JSP”(Java Server Pages) fortemente implementadas com “TagLibs”, do pacote “JSTL” ou do próprio “Struts”.

Para gerenciar o “lay-out” das páginas, deve-se utilizar o framework “Struts Tiles”, e para validação no “Struts Validator“.

3.3.1 Padrões de Interface

Um grupo de “Web Design“ definiu o padrão de interface que deve ser seguido pelas aplicações e que permita abordagens estéticas diferentes, mas que cuide da usabilidade e performance além de padronizar as aplicações. Se algo diferente for levantado, o grupo deve analisar e incorporar ao padrão se necessário.

3.3.2 Apresentação de Mensagens

Deve-se utilizar “TagLibs” para mostrar as mensagens que devem estar armazenadas no arquivo “applicationResources.properties”, juntamente com os textos fixos da aplicação(no caso de internacionalização).

3.4 Servidor de Aplicações

O servidor de aplicação utilizado é Jboss, devido ao seu completo suporte à plataforma JEE e sua robustez.

3.5 Sistema de Segurança

O sistema de segurança utilizado pelas aplicações é o sistema Sentinela, o qual é desenvolvido e mantido pela GIC.

3.6 Internacionalização

Caso um sistema necessite de internacionalização, todos os textos fixos (palavras ou frases que não mudam dinamicamente, tais como títulos, mensagens, etc) devem estar no arquivo “applicationResources.properties” e obtidos através deste, para que as traduções sejam facilitadas.

4 FERRAMENTAS

Ferramentas	Descrição	Fase	Disponibilidade
RSM - Rational Software Modeler	Ferramenta de modelagem	Análise/Projeto	12 cópias compartilhadas
Eclipse	IDE	Construção	livre
plugin MyEclipse	Ferramenta incorporada ao Eclipse com funcionalidades que melhoram a produtividade do desenvolvedor. Tem suporte para Struts, Hibernate, EJB, WYSIWYG editor, etc.	Construção	US$ 35,00 por ano, por desenvolvedor
Junit	Ferramenta de testes unitários	Construção	livre
Jmeter	Ferramenta de testes de stress	Construção	livre
Ant	Ferramenta de build	Construção	livre
CVS	Ferramenta de gerência de configuração	Todas	livre
DBDesigner	Ferramenta de Modelagem de Banco de Dados	Análise e construção	livre
iReport	Ferramenta de Relatórios	Construção	livre
Squirrel SQL	Ferramenta de acesso a Banco de Dados (Genérica – JDBC)	Construção, Análise e Projeto	livre
PostgreSQL	SGDB(Sistema Gerenciador de Banco de Dados)	Análise, Projeto e Construção	livre

5 AMBIENTE

5.1 Máquinas dos Desenvolvedores

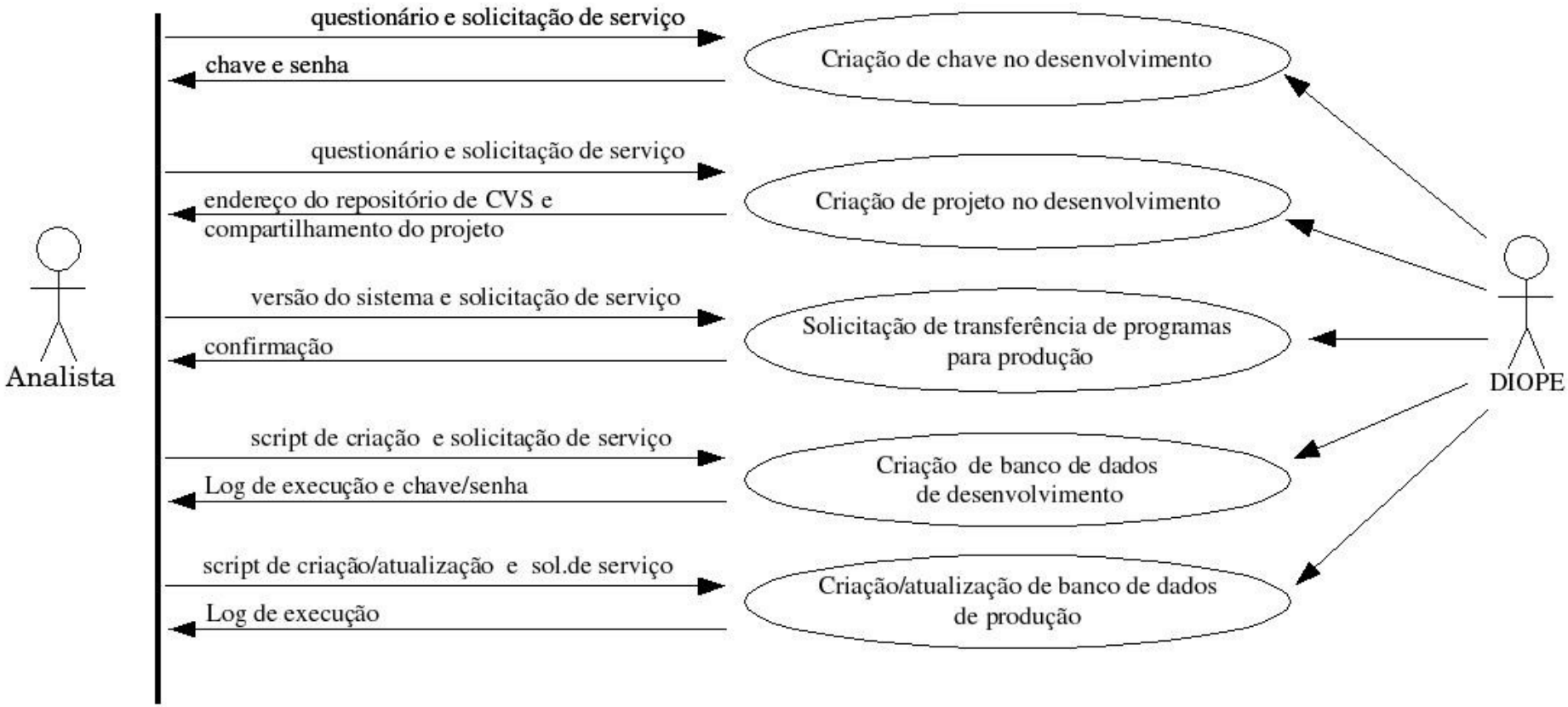
5.1.1 Hardware Mínimo

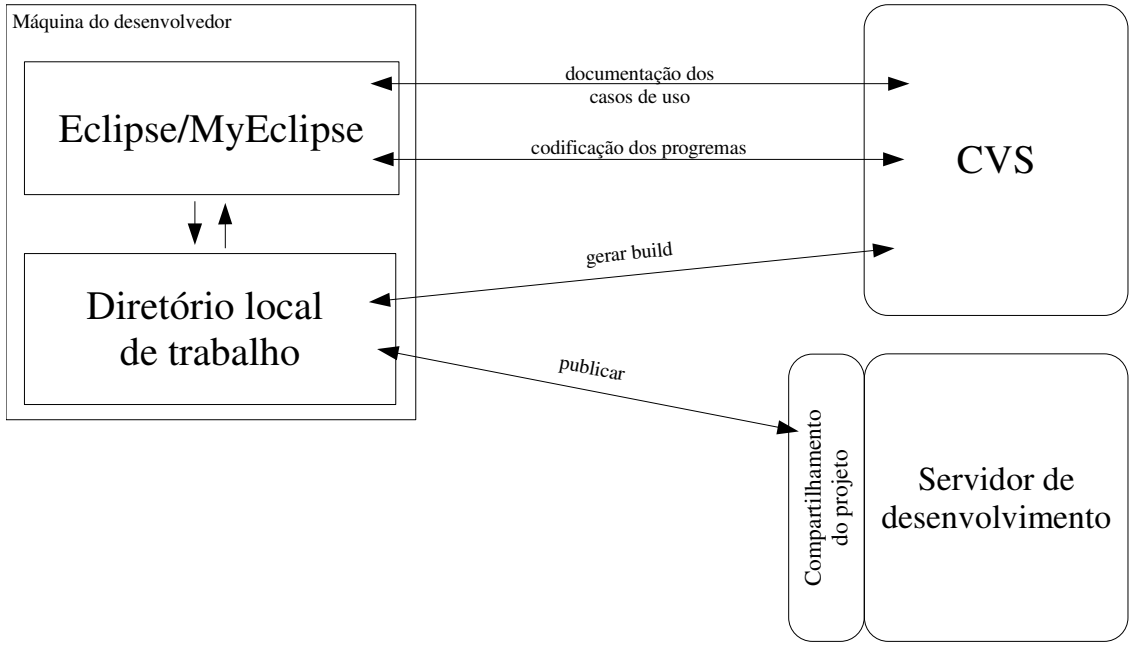
- processador de 1.8 ghz
- 512mb de memória RAM
- 20GB de Discos
- Monitor de 17”

5.1.2 Sistema Operacional

- GNU/Linux Debian

6 PROCESSOS DE AMBIENTE




Máquina do desenvolvedor
Eclipse/MyEclipse
Diretório local de trabalho
documentação dos casos de uso
codificação dos progremas
gerar build
publicar
CVS
Compartilhamento do projeto
Servidor de desenvolvimento