

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO
DO RIO GRANDE DO NORTE**

Jorge Guilherme Bezerra de Medeiros Júnior

**SISTEMA INTEGRADO DE GESTÃO EMPRESARIAL – ESTUDO DE CASO
DA CONIMPE ENGENHARIA**

**NATAL/RN
2011**

JORGE GUILHERME BEZERRA DE MEDEIROS JÚNIOR

**SISTEMA INTEGRADO DE GESTÃO EMPRESARIAL – ESTUDO DE CASO
DA CONIMPE ENGENHARIA**

Monografia do Curso de Graduação em
Sistemas de Informação da Faculdade
Natalense para o Desenvolvimento do Rio
Grande do Norte como requisito
obrigatório para obtenção do grau de
Bacharel.

Orientador:
Rômulo Fagundes Catanhede (Esp.)

Natal/RN
2011

Catálogo na Publicação – Biblioteca do UNI-RN
Setor de Processos Técnicos

Medeiros Júnior, Jorge Guilherme Bezerra de.

Sistema integrado de gestão empresarial – estudo de caso da Conimpe Engenharia / Jorge Guilherme Bezerra de Medeiros Júnior. – Natal, 2011.
45f.

Orientador: Prof. Rômulo Fagundes Catanhede (Esp.)

Monografia (Graduação em Sistemas de Informação) – Centro
Universitário do Rio Grande do Norte.

1. Sistema de Informação - Monografia. 2. Sistema Integrado de Gestão – Monografia. 3. Gestão Empresarial – Monografia. I. Catanhede, Rômulo Fagundes. II. Título.

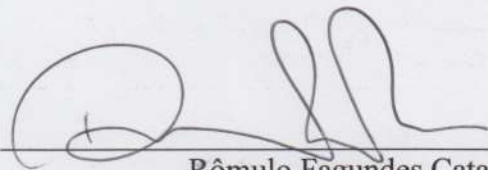
RN/UNI-RN/BC

CDU 004:658

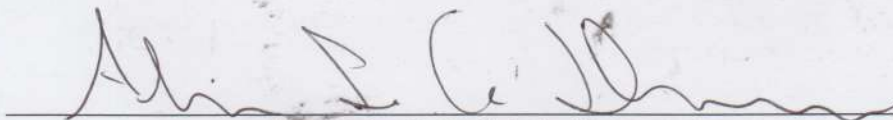
JORGE GUILHERME BEZERRA DE MEDEIROS JÚNIOR

**SISTEMA INTEGRADO DE GESTÃO EMPRESARIAL – ESTUDO DE CASO
DA CONIMPE ENGENHARIA**

Monografia do Curso de Sistemas de Informação da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte, () aprovado com a nota: ____ () não obtendo aprovação, em ____ de ____ de ____, avaliado pela banca examinadora composta pelos seguintes membros:



Rômulo Fagundes Catanhede
Professor Orientador



Alexandre Luiz Galvão Damasceno
Membro Examinador

Ricardo Wendell Rodrigues da Silveira
Membro Examinador

Dedico este trabalho a Dona
Fátima.

AGRADECIMENTOS

Agradeço a minha mãe Fátima e a minha esposa Cristiane.

Meus amigos Val Queiroz e Gustavo Sales pela força e oportunidade.

Agradeço ao meu professor e orientador Rômulo Fagundes pelo incentivo.

Agradeço ao meu colega de turma Fernando Lopes pelas inestimáveis dicas de PHP nas horas mais complicadas.

Agradeço aos amigos e colegas de turma, com os quais eu pude aprender ainda mais.

RESUMO

Sistemas integrados de gestão ou ERP (Enterprise Resource Planning) vem sendo amplamente utilizados nas empresas de grande porte de forma consolidada, no entanto, este recurso hora visto como vantagem estratégica pelas grandes corporações, hoje se tornou trivial, compondo a base de desenvolvimento da empresa, com a evolução e disseminação da tecnologia e sua consequente queda de preço, empresas consideradas de pequeno e médio porte se habilitam a implementar sistemas que gerenciem suas operações. ERPs são sistemas genéricos que tem o objetivo de integrar todas as informações dos diversos setores da empresa que são processadas através de um banco de dados único. Os resultados positivos e benefícios a serem obtidos com a adoção desses sistemas são bastante conhecidos, porém, as dificuldades a serem enfrentadas e a complexidade das mudanças a serem realizadas para a que os ganhos sejam reais não são tão perceptíveis para as empresas, principalmente para as de pequeno porte, que não possuem tantos recursos e até mesmo conhecimento gerencial para investimento em tecnologia. Este trabalho tem por objetivo relatar o resultado de uma pesquisa sobre a adoção de um sistema integrado de gestão, numa empresa de pequeno porte do ramo de engenharia civil, confrontando o referencial teórico e a prática empresarial.

Palavras chave: Sistemas de informação, Sistema Integrados de Gestão, Gestão Empresarial.

ABSTRACT

Integrated management systems or ERP (Enterprise Resource Planning) has been widely used in large companies on a consolidated basis, however, this feature has been seen as a strategic advantage by large corporations, has become trivial, forming the basis for development of the company with the evolution and spread of technology and the consequent drop in price, companies considered small and medium businesses to implement systems that enable them to manage their operations. ERP systems are generic which aims to integrate all the information from the various sectors of business that are processed through a single database. The positive results and benefits to be derived from the adoption of these systems are well known, however, the difficulties to be faced and complexity of changes to be made for the gains are real are not as noticeable for companies, especially for the small businesses that do not have many resources and even for knowledge management technology investment. This paper aims at reporting the results of a survey on the adoption of an integrated management system, a small business of the branch of civil engineering, comparing the theoretical and practical business.

Keywords: Information Systems, Enterprise Resource Planning, Business Management.

LISTA DE SIGLAS (Ordem Alfabética)

AJAX	Assincronous Javascript and XML
API	Application Programming Interface
CDDL	Common Development and Distribution License
CDK	Chemistry Development Kit
CSS	Cascading Style Sheets
DNS	Domain Name Server
EJB	Enterprise JavaBeans
EL	Expression Language
ER	Entidade Relacionamento
ERP	Enterprise Resource Planning
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IP	Internet Protocol
ISO	International Organization for Standardization
JBPM	Java Business Process Management
JDBC	Java Database Connectivity
JPA	Java Persistence API
JSF	Java Server Faces
JSLT	JavaServer Pages Standard Tag Library
JSP	Java Server Pages
ODBC	Open Data Base Connectivity
POJO	Plain Old Java Object
SIG	Sistema Integrado de Gestão
SQL	Structured Query Language
TCP	Transmission Control Protocol
UI	User Interface
UML	Unified Modeling Language
XHTML	eXtensible Hypertext Markup Language
XML	Extensible Markup Language

LISTA DE FIGURAS

Figura 1	Representação de um sistema.....	14
Figura 2	Arquitetura de uma aplicação Java.....	17
Figura 3	Tabela de percentual de uso de várias linguagens de programação.....	18
Figura 4	Ciclo de vida de uma aplicação JSF.	19
Figura 5	Componente <i>Calendar</i> do <i>Richfaces</i>	20
Figura 6	Fluxo de processamento da requisição usando <i>RichFaces</i>	21
Figura 7	Trecho de código usando <i>RichFaces</i>	22
Figura 8	Diagrama de componentes do <i>Facelets</i>	23
Figura 9	Componentes de uma aplicação que usa <i>JBoss Seam</i>	24
Figura 10	Arquitetura de objetos do <i>Hibernate</i>	27
Figura 11	Ciclo de vida do <i>Hibernate</i>	29
Figura 12	Relacionamento de tabelas de um banco de dados.....	30
Figura 13	Relacionamento N-N de tabelas de um banco de dados.....	31
Figura 14	Arquitetura do <i>Tomcat</i>	32
Figura 15	Tela inicial SIG Conimpe.....	36
Figura 16	Tela SIG Conimpe – Gerenciamento de Funcionários.....	38
Figura 17	Tela SIG Conimpe – Gerenciamento de Cargos.....	38

SUMÁRIO

1 INTRODUÇÃO.....	11
1.1 Objetivos.....	12
1.1.1 Objetivo Geral.....	12
1.1.2 Objetivos Específicos.....	12
1.2 Justificativa.....	12
2 REFERENCIAL TEÓRICO.....	14
2.1 Sistemas de Informação	14
2.2 Java 6	16
2.3 Java Server Faces	18
2.4 RichFaces	20
2.5 Facelets	22
2.6 Jboss Seam Framework	24
2.7 Hibernate	27
2.8 PostgreSql	31
2.9 Apache Tomcat 6.0	32
2.10 UML	34
2.11 Astah	34
2.12 Eclipse Helios IDE	34
3 METODOLOGIA.....	36
3.1 Tipo de pesquisa.....	36
3.1.1 SIG Conimpe.....	36
4 CONCLUSÕES E RECOMENDAÇÕES.....	40
REFERÊNCIAS.....	41
APÊNDICE.....	43
APÊNDICE 1 - Diagrama de Classes do Subsistema colaborativo do SIG Conimpe....	44
APÊNDICE 2 - Diagrama de Casos de Uso - SIG Conimpe (Colaborativo).....	45

1 INTRODUÇÃO

Potencializado pelo avanço tecnológico, é cada vez maior o custo-benefício e a necessidade de investimento nas empresas de pequeno e médio porte em tecnologia da informação. Aumentou também a pressão sobre as empresas para que acompanhem essas mudanças culturais em relação à tecnologia, o ritmo que a internet impôs as empresas tornou essa modernização um dos fatores fundamentais para o melhoramento da gestão e dos processos da empresa, resultando em vantagem estratégico-competitiva. Vantagens essas que podem ser a mudança e melhoramento dos processos, novas formas de interação com clientes, por exemplo. Wezel&Erdmann *apud* Vieira (2007) define Tecnologia da Informação como sendo um conjunto de *hardware* e *software* que desempenha uma ou mais tarefas de processamento de informações, como coletar, transmitir, estocar, recuperar, manipular e exibir dados. É uma ferramenta que reduz custos e agiliza a troca de informações, para isso assume várias funções dentro de uma empresa, tais como a automatização de processos, desenvolvimento de infraestruturas de comunicação interna e externa, melhoramento da comunicação entre a empresa e seus clientes e fornecedores e apoio na tomada de decisões. São poucas as atividades que escapam a 'onipresença' do computador e da internet, ou seja, são poucas também as atividades da empresa que não estão diretamente ligadas a tecnologia.

O ERP pode ser definido como um conjunto integrado de sistemas, com banco de dados único que proporciona a integração de setores que, mesmo atuando em áreas distintas, podem operar integrados, gerando um maior fluxo de informações na empresa, como finanças, recursos humanos, estoque, compras e vendas. Em termos gerais, trata-se de uma plataforma de *software* desenvolvida para integrar os diversos departamentos de uma empresa, sua adoção põe fim às aplicações redundantes (que em alguns casos se restringe a software do tipo Pacote do *Office* da Microsoft) e incompatíveis existentes nas empresas possibilitando a automação, armazenamento, padronização dos processos e regras de negócio, geração de relatórios de todas as informações da empresa.

A Conimpe Engenharia está presente no mercado da construção civil industrial e residencial do Rio Grande do Norte desde 1990. A empresa destaca-se no segmento da engenharia civil aplicada a indústria eletromecânica, montagem e manutenção industrial, com ênfase nas áreas de petróleo e gás. A empresa possui em seu portfólio, serviços nas áreas técnicas de desenvolvimento de projetos e obras civis; elétricas e mecânicas. Contando com uma equipe de mais de 45(quarenta e cinco) funcionários.

O uso estratégico da tecnologia da informação trará sem dúvidas facilidade para elaborar e alcançar objetivos de forma mais sólida, ajudando a direção da Conimpe administrar de maneira integrada, compartilhar conhecimento, transmitir e aprimorar habilidades além de reduzir as perdas geradas pela distância e pelo tempo. Com a popularização da internet banda larga, tornou-se possível que aplicações robustas sejam executadas de um servidor remoto, reduzindo o investimento em hardware, em locais apropriados para acomodar um servidor, em manutenção e treinamento de funcionários especializados em gerenciamento de servidores, dentre tantos outros que poderiam ser citados. Um sistema de gerenciamento que se adeque as necessidades específicas de uma empresa contribui para o aumento do desempenho dos setores, melhorando o gerenciamento dos processos da empresa.

Diante do exposto, esta pesquisa evidencia a necessidade de ampliar os conhecimentos a respeito de como um sistema integrado de gestão pode otimizar os processos de uma empresa.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver um sistema de informação que integre os setores operacional, de recursos humanos, projetos, grupos de trabalho, e gere informação que ajude nas tomadas de decisões gerenciais em tempo real da Conimpe Engenharia.

1.1.2 Objetivos Específicos

- Analisar as principais necessidades de otimização de processos operacionais, de recursos humanos e colaborativos.
- Elaborar artefatos de modelagem do sistema (UML).
- Elaborar diagrama E-R (Entidade-Relacionamento).
- Documentar os diversos módulos do sistema.
- Codificar o sistema em linguagem de programação Java.
- Implementar o banco de dados da aplicação.

1.2 Justificativa

O mercado de pequenas e médias empresas do ramo de engenharia civil cada vez mais competitivo, que exige cada vez mais uma gestão eficiente toda possibilidade de vantagem

estratégica deve ser explorado pelas empresas, e um dos pontos fracos que se observa são os sistemas de gestão dessas empresas, por serem em sua maioria administradas por engenheiros, onde o lado técnico é priorizado, a gestão da empresa por vezes é deixada em segundo plano, a partir disso surge a necessidade da elaboração de um sistema de gestão de recursos e processos, que pode gerar grandes melhorias de desempenho e qualidade, garantindo assim uma vantagem estratégica sobre empresas que possuam um desempenho gerencial e operacional inferior.

Hoje na Conimpe Engenharia se observa com precisão a situação descrita anteriormente, os profissionais são totalmente voltados para o lado técnico, deixando o lado gerencial a desejar, os processos são controlados em sua maioria por arquivos do tipo Pacote do *Office* da Microsoft, como processadores de texto (*Word*), planilhas eletrônicas (*Excel*), não existem, por exemplo, um controle de estoque ou gerenciamento de tarefas para os funcionários.

Este trabalho busca elaborar um sistema de gestão de recursos para a Conimpe Engenharia, uma vez que a empresa busca melhorar seu desempenho através da melhoria da gestão da informação na empresa.

2 REFERENCIAL TEÓRICO

2.1 Sistemas de Informação

Segundo O'Brien (2001), um sistema pode ser definido como “um grupo de componentes inter-relacionados que trabalham juntos rumo a uma meta comum recebendo insumos e produzindo resultados em um processo organizado de transformação”. Para o autor, um sistema (FIGURA 1) possui três funções básicas que interagem entre si. São elas:

Entrada: Que tem como função receber elementos que serão processados pelo sistema.

Processamento: Envolve processos de transformação dos elementos recebidos na entrada em produto final.

Saída: Tem como objetivo transferir os elementos produzidos no processamento até o seu destino final.

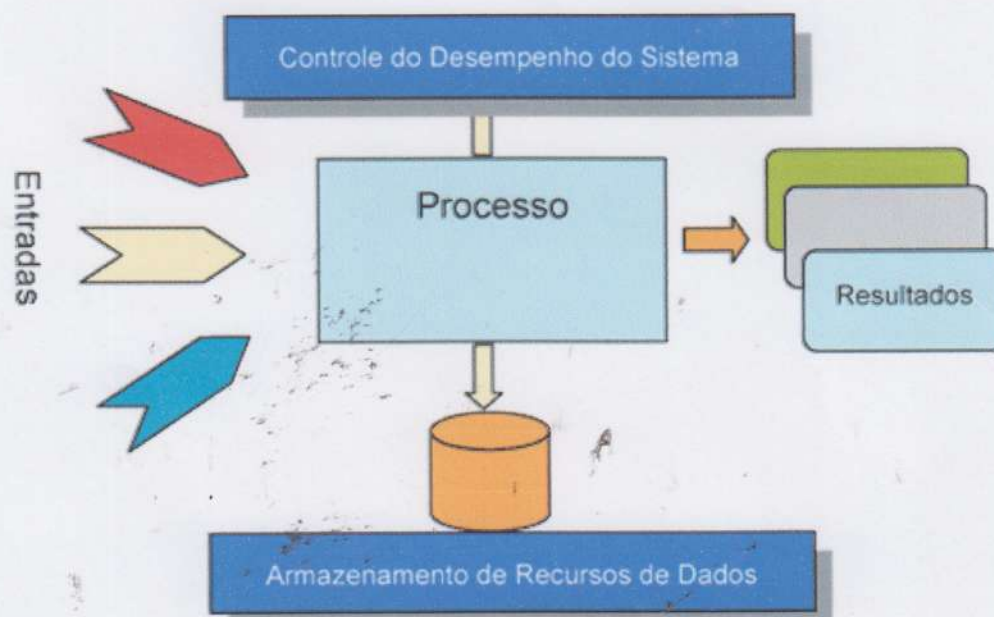


Figura 1: Representação de um sistema.

Fonte: <http://www.scielo.br/img/revistas/jistm/v7n1/06f1.gif>

Uma ampliação do conceito de sistema passa pela inclusão de dois componentes adicionais: *feedback* e controle. Um sistema que os possua pode ser considerado automonitorado ou autorregulado.

Feedback: São dados sobre o desempenho do sistema.

Controle: É a função que monitora e avalia os *feedbacks* para determinar se um sistema está trabalhando corretamente para a obtenção de um objetivo. Também se encarrega de

ajustar componentes de entrada e processamento de um sistema para garantir que metas sejam cumpridas.

Existem ainda algumas outras características que são importantes para uma compreensão mais completa de um sistema de informação. Um sistema não existe isoladamente, ele existe e funciona em um ambiente que pode conter outros sistemas. Se o sistema for um dos componentes de um sistema maior, ele é considerado um subsistema, e o sistema principal é o seu ambiente.

Transpondo o conceito genérico de sistemas para o universo dos sistemas de informação, ainda segundo O'Brien (2001) ele é composto por cinco recursos principais:

Recursos Humanos:

Especialistas – analistas de sistemas, programadores e operadores de computador.

Usuários Finais – todos os usuários que utilizam o sistema.

Recursos de Hardware:

Máquinas – computadores, monitores de vídeo, unidades de armazenamento de dados, impressoras.

Mídias – cd, dvd, pendrive.

Recursos de Software:

Programas – sistemas operacionais, browsers.

Procedimentos – procedimento de entrada de dados, procedimentos de correção de erros.

Recursos de Dados:

Banco de dados, cadastro de clientes, arquivos de funcionários, descrição de produtos.

Recursos de Rede:

Meios de comunicação, acesso aos softwares de gestão e comunicação.

Produtos de Informação:

Relatórios administrativos, que utilizem textos e gráficos, informação em tempo real.

Os sistemas de informação podem ser divididos e classificados em tipos, dois são considerados principais, sistemas de apoio às operações e sistemas de apoio a tomada de decisão. Estas classificações destacam os papéis que cada um desses sistemas desempenha nas operações e na administração dos negócios.

Sistemas de Apoio as Operações:

São responsáveis pela produção de informação no nível operacional de uma empresa, seu enfoque está nos processos transacionais e industriais, bem como apoiar a comunicação e colaboração e atualizar o banco de dados da empresa. Um subsistema de processamento de transações registra e processa dados resultantes das ações comerciais da empresa. Já os subsistemas de controle de processos monitoram e controlam as ações físicas da empresa. Por fim, os subsistemas colaborativos atuam no aumento da comunicação entre os diversos setores e colaboradores da empresa, seu objetivo é o aumento da produtividade de equipes e grupos de trabalho.

Sistemas de Apoio Gerencial:

São sistemas quem tem o objetivo de fornecer informação precisa que ajude na tomada de decisões eficazes nas estratégias de gestão, essas informações devem ser dispostas na forma de relatórios e gráficos. De acordo com Caiçara Júnior (2008) um sistema de apoio gerencial pode constituir desde uma simples planilha eletrônica de custos de produção até um sistema grande e complexo, como simuladores de um ambiente mercadológico ou industrial, com controle de mão-de-obra, máquinas, insumos, custos, etc.

Ainda segundo Caiçara Júnior (2008), um dos recursos mais utilizados nesse tipo de sistema é a análise de simulações de cenários. As entradas de dados podem ser de fontes internas ou externas, a partir de outro sistema integrado de gestão, ou fontes externas, como dados governamentais ou de concorrentes.

Uma aplicação de sistema de apoio gerencial possui um alto uso de processamento, recursos avançados de bancos de dados e um banco de modelos responsável por analisar os dados através de modelos financeiros ou estatísticos ou, ainda, através da geração de gráficos e tabelas de forma dinâmica.

2.2 Java 6

Segundo Deitel&Deitel (2001), Java é uma linguagem de programação inicialmente desenvolvida por volta de 1991 e lançada no mercado para uso em 1995, pela empresa Sun Microsystems, possui sintaxe semelhante as linguagens C e C++. Por ser uma linguagem orientada a objetos, Java usa os benefícios de reutilização de código, extensibilidade e aplicações dinâmicas.

Ainda segundo os autores, programas Java consistem em partes chamadas 'classes'. Estas consistem em outras partes chamadas de 'métodos' que realizam tarefas e retornam as informações ao completarem suas tarefas. Além de poder programar cada uma das classes que são necessárias à implementação de um sistema, pode-se também utilizar as muitas coleções de classes existentes em bibliotecas de classes Java, estas bibliotecas são chamadas de Java APIs.

De acordo com Tacla (2007), por ser uma linguagem que segue o paradigma da orientação a objeto um *software* Java é estruturado da seguinte forma (FIGURA 2):

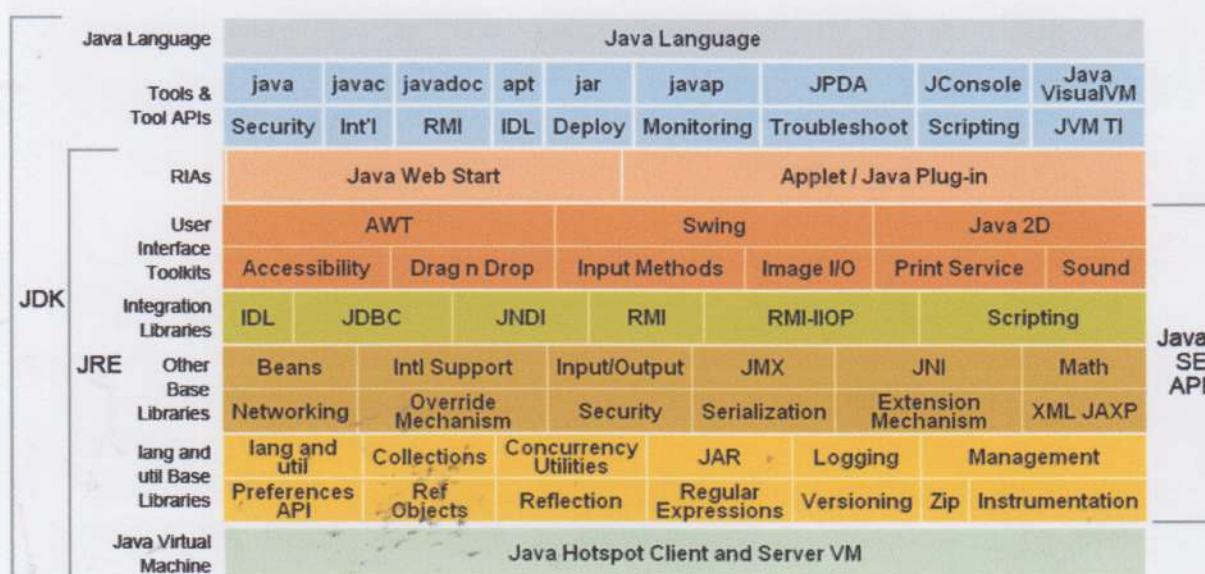


Figura 2: Arquitetura de uma aplicação Java.

Fonte: <http://javafree.uol.com.br/dependencias/tutoriais/javatutorial/j2se5.jpg>

Objeto: É uma abstração que representa uma entidade do mundo real, pode ser algo concreto ou abstrato e possui 3 (três) propriedades:

Estado: Definido pelo conjunto de propriedades do objeto(atributos) e de suas relações com os outros objetos.

Comportamento: Como um objeto responde às solicitações dos outros, corresponde a todas as ações possíveis de um objeto.

Identidade: Significa que cada objeto é único no sistema.

Classe: É um conjunto de objetos com propriedades, comportamento, relacionamentos e semântica comum. Uma classe pode ser vista como um modelo para criar objetos.

Segundo a Tiobe Software¹, Java vem se mantendo como a linguagem de programação mais utilizada para o desenvolvimento de novas aplicações (FIGURA 3).

Position Dec 2011	Position Dec 2010	Delta in Position	Programming Language	Ratings Dec 2011	Delta Dec 2010	Status
1	1	=	Java	17.561%	-0.44%	A
2	2	=	C	17.057%	+0.98%	A
3	3	=	C++	8.252%	-0.76%	A
4	5	↑	C#	8.205%	+1.52%	A
5	8	↑↑↑	Objective-C	6.805%	+3.56%	A
6	4	↓↓	PHP	6.001%	-1.51%	A
7	7	=	(Visual) Basic	4.757%	-0.36%	A
8	6	↓↓	Python	3.492%	-2.99%	A
9	9	=	Perl	2.472%	+0.14%	A
10	12	↑↑	JavaScript	2.199%	+0.69%	A
11	11	=	Ruby	1.494%	-0.29%	A
12	10	↓↓	Delphi/Object Pascal	1.245%	-0.93%	A
13	13	=	Lisp	1.175%	+0.11%	A
14	23	↑↑↑↑↑↑↑	PL/SQL	0.803%	+0.24%	A
15	14	↓	Transact-SQL	0.746%	-0.03%	A
16	16	=	Pascal	0.734%	-0.03%	A
17	18	↑	Ada	0.632%	-0.02%	B
18	35	↑↑↑↑↑↑↑	Logo	0.619%	+0.26%	B
19	17	↓↓	Assembly	0.563%	-0.10%	B
20	25	↑↑↑↑	ABAP	0.560%	+0.01%	B

Figura 3: Tabela de percentual de uso de várias linguagens de programação.

Fonte: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

2.3 JavaServer Faces

Segundo Geary & Horstmann (2007), o *JavaServer Faces* pode ser definido como um *framework* da camada de visualização de uma aplicação *Java Web*, é responsável pela interação do usuário com o sistema. Visando o aumento de produtividade no desenvolvimento de componentes de interface reutilizáveis e padronizados. Já segundo Mann (2004), o *JavaServer Faces* define uma forma comum de construir *widgets* UI (botões, *links*, *selects*, caixas de texto). Outro ponto importante é o controle do fluxo de dados da aplicação bem como o armazenamento dos estados das requisições. As funções básicas do JSF são:

¹ <http://www.tiobe.com>

processamento de formulários, tratamento de eventos, tratamento de erros, conversão de tipos, internacionalização e validação de dados, possuem um ciclo de vida onde todas essas funções são executadas até o resultado final. A especificação JSF definida pelos autores Geary & Horstmann (2007) possui 6 (seis) fases distintas (FIGURA 4):

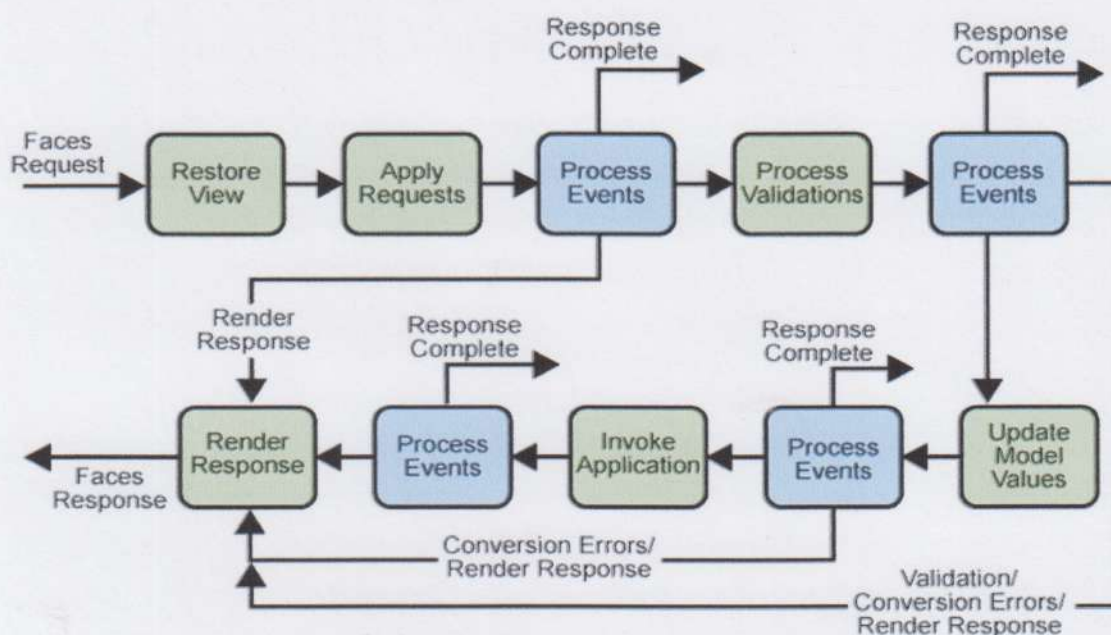


Figura 4: Ciclo de vida de uma aplicação JSF.

Fonte: <http://docs.oracle.com/javaee/5/tutorial/doc/figures/jsfIntro-lifecycle.gif>

1. Restore view

Se a página já foi exibida antes, é recuperada toda a árvore de componentes para a página requisitada. Caso esteja sendo exibida pela primeira vez é construída uma nova árvore de componentes. No caso da página já ter sido exibida antes, todos os componentes serão configurados com seu estado anterior. É fácil notar isso no processo de validação, onde os valores digitados inicialmente no formulário serão mantidos. Se a requisição não possuir dados solicitados, a implementação JSF passa para a fase *Render Response*. Normalmente, isso acontece quando a página é requisitada pela primeira vez.

2. Apply request values

Nesta fase ocorrem iterações sobre os objetos da árvore de componentes. São extraídos todos os valores digitados pelo usuário e guardados nos seus respectivos componentes.

3. Process validation

A *String* com o valor enviado é convertida para o tipo correto do objeto. Caso ocorra algum erro de validação, uma mensagem de erro é adicionada no *FacesContext*, então o componente é marcado como inválido e a implementação JSF invoca a fase *Render Response*. Caso contrário o ciclo de vida continua normalmente.

4. Update Model Values

Sua principal atividade é atribuir os valores informados pelo usuário no formulário, para as respectivas propriedades associadas aos *BackBeans*, também conhecidos como *Managed Beans*. Nesta fase do ciclo de vida pode haver erro na conversão, fazendo com que o JSF dispare um erro de tempo de execução.

5. Invoke Application

Nesta fase o controlador (*controller*) do JSF chama o método associado no *submit* (envio da requisição ao servidor), disparando assim a camada de regras de negócio da aplicação. Todos os valores foram validados e carregados nas fases anteriores, por isso podem ser utilizados conforme a necessidade do desenvolvedor. Geralmente, é retornada uma *String* de resultado, que é passada para o handle de navegação.

6. Render Response

Esta fase codifica a resposta e envia-a de volta ao navegador.

2.4 RichFaces

Segundo o desenvolvedor do *richfaces* ele é um *framework* que adiciona capacidades de AJAX em uma aplicação que usa JSF, adicionando algumas facilidades ao desenvolvedor, como paginação de resultados, janela modal, calendário, etc. (FIGURA 5).

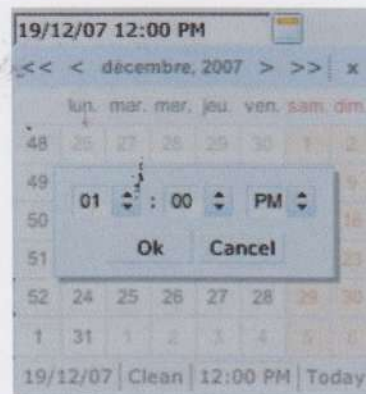


Figura 5: Componente Calendar do Richfaces.

Fonte: O Autor.

Os componentes desta biblioteca possuem um incrível suporte AJAX, e ela, pode ser considerada uma extensão do *Ajax4jsf* com inúmeros componentes com AJAX “embutidos” e com um suporte a *Skins* que podem deixar as interfaces da aplicação com um visual padronizado. (SOUZA, 2011)

De acordo com Katz (2008) o *RichFaces* é um *framework opensource* que incorpora AJAX em aplicações JSF sem recorrer ao *javascript*.

Permite integrar capacidades AJAX no desenvolvimento do negócio da aplicação, e como os componentes *RichFaces* são disponibilizados prontos para a utilização, é possível tirar rapidamente, proveito de todas as suas funcionalidades, para criar aplicações *web* que permitam uma interação entre o utilizador e aplicação mais agradável e rápida.

Este *framework* se diferencia de outros baseados em AJAX uma vez que, por conter suporte orientado à página, ao contrário da tradicional orientação ao componente, consegue-se definir eventos AJAX numa página e também quais as áreas da página que precisam ser sincronizadas com o componente JSF *Tree* depois desse evento AJAX ser processado no servidor (FIGURA 6 e 7).

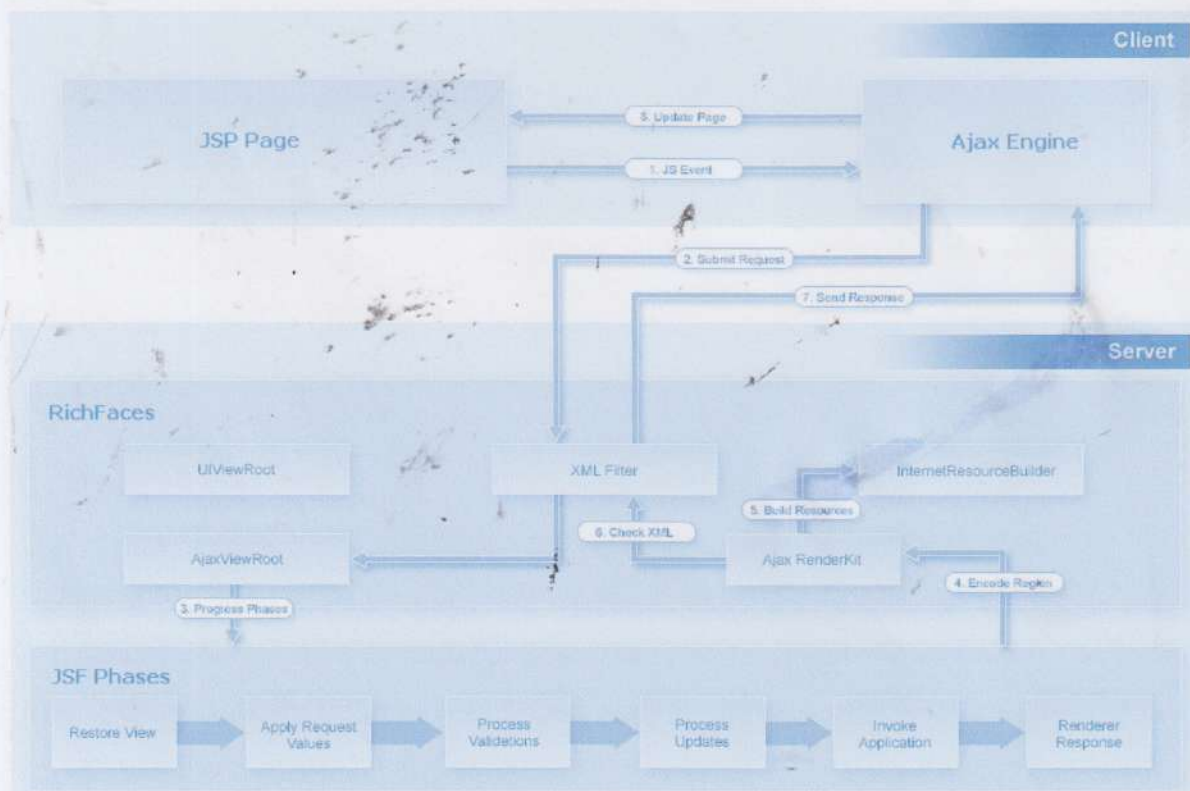


Figura 6: Fluxo de processamento da requisição usando RichFaces.

Fonte: <http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/RichFaces?xpage=print&>

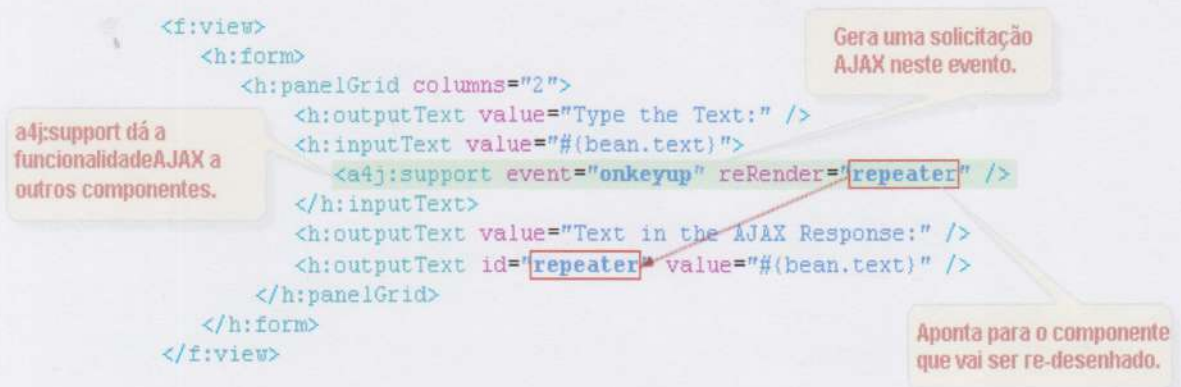


Figura 7: Trecho de código usando RichFaces.

Fonte: <http://pptinformatica.com.br/wordpress/wp-content/uploads/2010/02/listagem01.gif>

De forma sintetizada as inúmeras vantagens do *RichFaces* são listadas em seguida:

- *RichFaces* é totalmente integrado ao JSF e seu ciclo de vida, e permite invocar do lado do servidor processos de validação e conversão durante o ciclo de reposta ao pedido AJAX.
- Maior flexibilidade no aspecto visual, uma vez que se baseia em componentes *skinnable*, permitindo que seja dada mais importância ao negócio da aplicação.
- Um vasto número de componentes que permitem uma maior flexibilidade no desenvolvimento de aplicações *web* assim como poderosas funcionalidades com componentes *out-of-the-box* (prontas para usar).
- Capacidade de criar novos componentes com suporte AJAX, através do *Component Development Kit* (CDK) que inclui facilidades na geração de código e *templates* usando sintaxe JSP. Com isto é possível reduzir ou eliminar rotinas na criação de componentes, permitindo menos esforço e tempo no seu desenvolvimento.
- Permite também suporte avançado de gestão a diferentes tipos de recursos *javascript* e *stylesheets* CSS.
- Esta *framework* gera automaticamente casos de teste à medida que os componentes são desenvolvidos, não testando apenas os componentes, mas sim outras funcionalidades do lado do cliente ou servidor incluindo código *javascript*.

2.5 Facelets

Segundo Aranda & Wadia (2008), o *Facelets* é uma linguagem de *templating* desenvolvida para trabalhar em conjunto com JSF. Possui vantagens que facilitam a criação e

reutilização de páginas e componentes (FIGURA 8), devido ao seu mecanismo de *templates*, por não usar a *engine* do JSP, páginas que usam *facelets* são aproximadamente 30% mais rápidas que as JSPs.

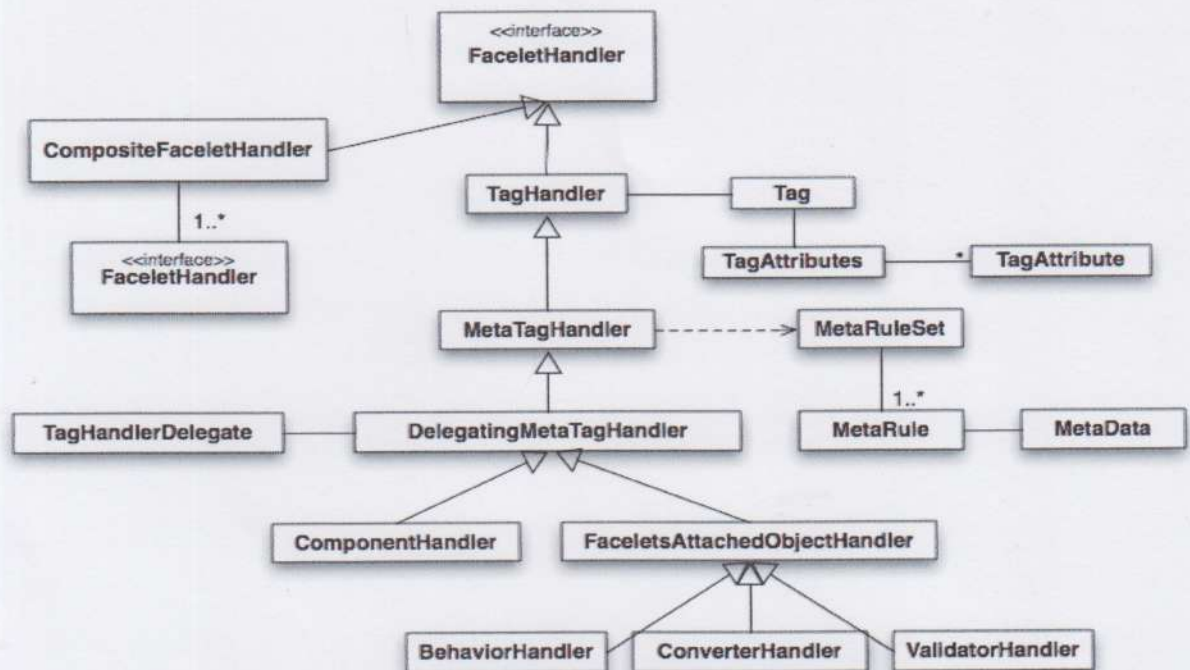


Figura 8: Diagrama de componentes do Facelets.

Fonte: <http://javaserverfaces.java.net/nonav/docs/2.1/javadocs/javax/faces/view/facelets/>

Segundo Paiva *et al* (2011), o *Facelets* é um subprojeto do JSF mantido pela Sun. Ele é facilmente integrado ao JSF e suas principais características são:

- É um *Framework* de código aberto e uso livre, sob licença CDDL 1.0;
- Funciona em conjunto com as especificações JSF 1.1 e JSF 1.2, incluindo a RI da Sun e o *Apache MyFaces*;
- Adicionado como componente padrão nas versões superiores da JSF;
- Possibilidade de especificar a hierarquia de componentes de arquivos separados (*UICompositions*);
- Relatório preciso do Erro (Linha, *Tag* e Atributo);
- Especifica *Tags* em arquivos separados ou até mesmo em *Jars* Separados;
- Suporte completo a EL, incluindo funções.
- Foi criado especificamente para JSF, resolvendo deficiências encontradas nas versões iniciais;

Ainda segundo os autores as funcionalidades principais do *Facelets* são:

- Disponibilizar um novo compilador de páginas baseado em XHTML;
- Realizar a criação da árvore de componentes das telas JSF;
- Suporte as *custom tags*: JSF Core, JSF HTML e JSTL;
- Criação de *templates* de tela JSF e reuso (herança) de telas;
- Composição de componentes JSF para a formação de novos componentes;
- Definição de *custom tags* para usar os novos componentes visuais.

2.6 Jboss Seam Framework

De acordo com Allen (2008) o JBoss Seam é um *framework* que integra tecnologias como *Asynchronous JavaScript and XML (AJAX)*, *JavaServer Faces (JSF)*, *Java Persistence API (JPA)*, *Enterprise Java Beans (EJB 3.0)* e *Business Process Management*. Foi desenvolvido para eliminar a complexidade em níveis de arquitetura da aplicação e oferecer aos desenvolvedores um total controle sobre a implementação da lógica de negócio sem se preocupar com a exposição das informações ou configuração excessiva de arquivos XML, dispondo de anotações para classes Java e componentes bem definidos para a camada de apresentação. (FIGURA 9)

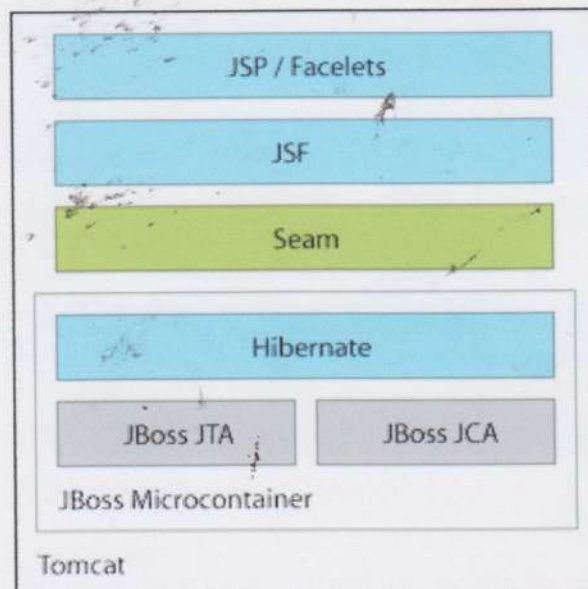


Figura 9: Componentes de uma aplicação que usa JBoss Seam.

Fonte: <http://docs.jboss.org/seam/1.0.1.GA/reference/en/html/configuration.html>

Segundo Farley (2007), o *Seam* segue alguns conceitos básicos de desenvolvimento:

- **DataModel:** Entidades (*Entity*) devem representar os objetos de negócio a serem persistidos e utilizados nas regras de negócio. Podem-se utilizar entidades gerenciadas via JPA ou POJOs (*Plain Old Java Objects*) mapeados no *Hibernate*;
- **User Interface(UI):** As páginas JSF exibem a interface gráfica, capturando os dados enviados via formulário e mostrando os resultados. Os campos, na interface gráfica, são mapeados aos objetos de negócio, acessado via JSF com EL (*Expression Language*);
- **Handle Web Events:** A camada de negócio pode ser implementada usando EJB3 *session beans* ou apenas classes Java com anotações. Esta camada atuará como *controller* para eventos disparados pelas páginas JSF.

Ainda segundo Farley (2007) o *JBoss Seam Framework* possui vários escopos de atuação, também chamados de contextos de tempo de execução, as informações podem ser passadas para vários contextos em tempo de aplicação, que tem um ciclo de vida definido pelo contexto definido no escopo.

De acordo com Carmo e Freire (2009) o *Seam* possui os seguintes contextos:

- **Stateless:** Contexto para armazenar a referência para objetos *stateless*. Tecnicamente não é um contexto já que cada busca ou chamada a método num objeto resulta numa nova referência para o *bean*. Este *bean* é gerenciado pelo *pool* de *beans* do *EJB Container*.
- **Request:** Objetos armazenados neste escopo tem o tempo de vida de uma requisição *Web* realizada por um usuário através do *browser*. Este escopo passa a existir no momento que o servidor atende a requisição e será destruído ao final da requisição *Web* quando o conteúdo gerado for enviado para o *browser*. Este contexto permite armazenar objetos que podem ser passados de um *servlet* para outro ou de um *servlet* para uma página JSP se ocorrer um encaminhamento (*forward*) de um componente para outro enquanto a requisição ainda está em processamento no lado do servidor. Pode-se imaginar este contexto como o escopo determinado pela passagem de parâmetros quando um objeto invoca o método de outro objeto.

- **Page:** Objetos armazenados neste escopo tem o tempo de vida de apenas o tempo de execução (processamento) de uma página JSP no lado do servidor. Após a geração do conteúdo HTML pelo JSP no lado do servidor e o envio deste conteúdo para o *browser*, as referências para os objetos armazenados neste contexto são perdidos. Podemos imaginar este contexto como o escopo de um método onde declaramos variáveis locais que são destruídas ao final da execução do método.
- **Conversation:** Este contexto permite armazenar objetos que terão um tempo de vida maior que uma requisição (*event*) e menor que uma sessão (*session*). Com este contexto se torna possível definir um escopo para objetos que são usados para implementar um fluxo de telas que realizam um caso de uso. Em um *Seam Component* é possível anotar o início da conversação num método e anotar outro para demarcar o fim da conversação. Este contexto permite abrir várias janelas de *browser* para uma mesma tela JSF e cada janela representar uma conversação diferente (contexto diferente de objetos) para execução simultânea do mesmo caso de uso. Cada janela é um contexto separado que não influencia outro contexto aberto.
- **Session:** Este contexto define um escopo para manter objetos como, por exemplo, o usuário autenticado ou outros objetos a serem compartilhados entre várias conversações (escopo de conversação), Além de objetos globais do usuário que acessa a aplicação.
- **Application:** Objetos armazenados neste contexto estarão disponíveis para todas as conversações e usuários que acessam a aplicação. Este contexto é útil ao armazenar objetos com conteúdo que não mudam com frequência para compartilhar para toda aplicação como, por exemplo, configurações ou meta informações. Muitos objetos internos do próprio *Seam* são armazenados neste contexto.
- **Business Process:** Define um contexto para objetos que são associados a um processo de negócio gerenciado pelo jBPM. Objetos armazenados neste contexto pertencem a um processo e podem ser compartilhados entre vários usuários que acessam o mesmo processo de negócio. Só faz sentido usar este escopo se a máquina de processos implementada pelo jBPM estiver em uso numa aplicação *Web* em conjunto com o *Seam*.

2.7 Hibernate

Segundo Elliott, O'Brien e Fowler (2009), o *Hibernate* é um *framework* que implementa o mapeamento objeto relacional para Java, que consiste em associar classes Java a tabelas em um banco de dados relacional e disponibilizar de forma transparente consultas ao banco de dados. É uma forma mais fácil e produtiva de trabalhar com bancos de dados relacionais na forma de objetos Java, já que o uso do *framework* evita que o desenvolvedor tenha que escrever código SQL para as consultas necessárias ao banco de dados.

De acordo com a JBoss Community² que desenvolve o *Hibernate*, ele possui uma arquitetura flexível formada basicamente por interfaces, suportando vários tipos de abordagens. Alguns dos principais objetos do *Hibernate* são descritos na FIGURA 10:

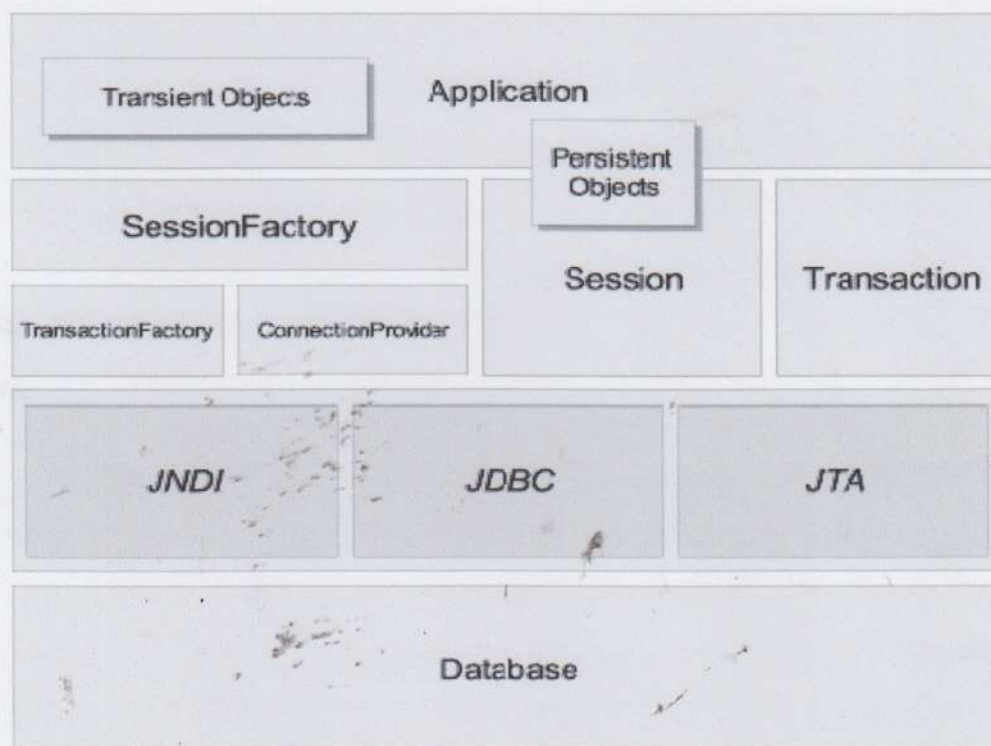


Figura 10: Arquitetura de objetos do Hibernate.

Fonte: GALHARDO e LIMA, 2007.

- **SessionFactory:** Um objeto do tipo *SessionFactory* mantém o mapeamento objeto relacional em memória. Permitindo a criação de objetos do tipo *Session*, a partir dos quais os dados são acessados, também denominados como fábrica de objetos *Sessions*.

² <http://www.jboss.org/>

- **Session:** Um objeto do tipo *Session* possibilita a comunicação entre a aplicação e a persistência, através de uma conexão JDBC. É um objeto leve de ser criado, não deve ter tempo de vida por toda a aplicação. Um objeto *Session* possui um *cache* local de objetos recuperados na sessão. Com ele é possível criar, remover, atualizar e recuperar objetos persistentes.
- **Configuration:** O objeto *Configuration* é utilizado para realizar as configurações de inicialização do *Framework*. Através dele são definidas as diversas configurações do *Hibernate*, como o driver do banco de dados a ser implementado, o dialeto, o usuário e senha do banco, entre outras. É a partir de uma instância desse objeto que se indica como os mapeamentos entre classes e tabelas de banco de dados devem ser feitos.
- **Transaction:** A interface *Transaction* é utilizada para representar uma unidade indivisível de uma operação de manipulação de dados. O uso dessa interface em aplicações que usam *Hibernate* é opcional. Essa interface abstrai a aplicação dos detalhes das transações JDBC.

Já segundo Galhardo e Lima (2007), em aplicações orientadas a objetos, a persistência permite que um objeto continue a existir mesmo após a destruição do processo que o criou. Na verdade, o que continua a existir é seu estado, já que pode ser armazenado em disco e então, no futuro, ser recriado em um novo objeto. Os objetos do *Hibernate* possuem 3 (três) estados de instância: (FIGURA 11)

- **Transientes:** são objetos que suas instâncias não estão nem estiveram associados a algum contexto persistente. Eles são instanciados, utilizados e após a sua destruição não podem ser reconstruídos automaticamente;
- **Persistentes:** são objetos que suas instâncias estão associadas a um contexto persistente, ou seja, tem uma identidade de banco de dados.
- **Detached:** são objetos que tiveram suas instâncias associadas a um contexto persistente, mas que por algum motivo deixaram de ser associadas, por exemplo, por fechamento de sessão, finalização de sessão. São objetos em um estado intermediário, nem são transientes nem persistentes.

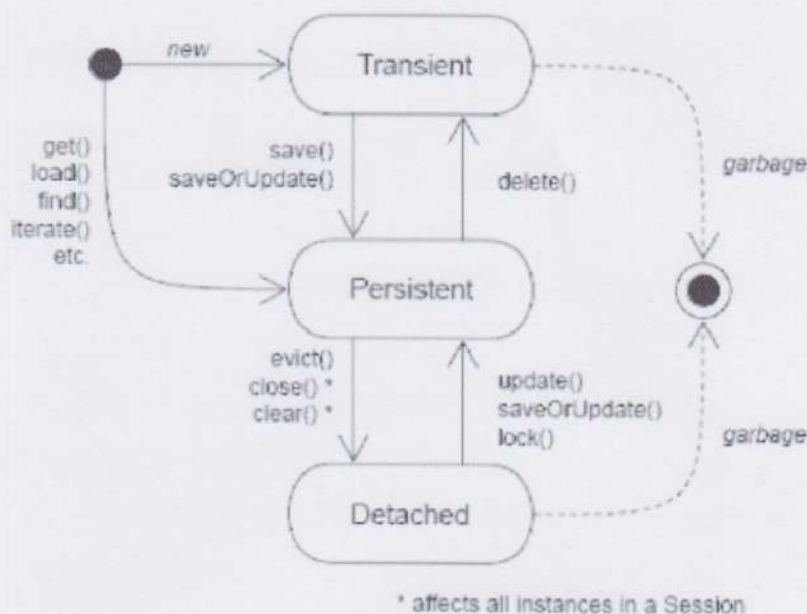


Figura 11: Ciclo de vida do Hibernate.

Fonte: GALHARDO e LIMA, 2007.

Ainda segundo Galhardo e Lima (2007) o *Hibernate* utiliza objetos *Session* para persistir e recuperar objetos. Um objeto *Session* pode ser considerado como uma sessão de comunicação com o banco de dados através de uma conexão JDBC.

Os autores afirmam que o mapeamento objeto-relacional do *Hibernate* pode ser feito de duas formas, usando arquivo do tipo XML, ou através de anotações, que podem ser definidas como metadados que aparecem no código fonte e são ignorados pelo compilador. Qualquer símbolo em um código Java que comece com uma arroba (@) é uma anotação. Este recurso foi introduzido na linguagem Java a partir da versão Java SE 5.0. Em outras palavras, as anotações marcam partes de objetos de forma que tenham algum significado especial.

O termo associação é utilizado para se referir aos relacionamentos entre as entidades. Os relacionamentos *n-para-n*, *n-para-1* e *1-para-n* são os mais comuns entre as entidades de um banco de dados. (GALHARDO e LIMA, 2007)

Os principais relacionamentos são:

- **1-n (one-to-many):** Para exemplificar o relacionamento 1-n, considera-se o relacionamento entre a entidade Centro e a entidade Universidade (FIGURA 12). O relacionamento diz que uma universidade possui um conjunto de n centros e um centro está associado a apenas uma única universidade.

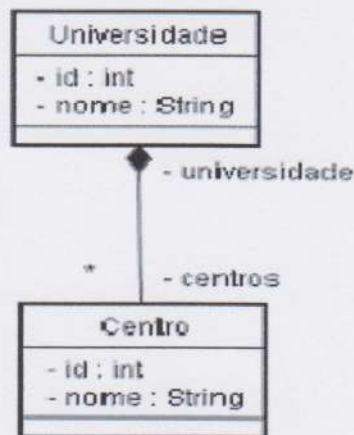


Figura 12: Relacionamento de tabelas de um banco de dados.
Fonte: GALHARDO e LIMA, 2007.

- **n-1 (many-to-one):** O relacionamento n-1 será apresentado a partir do relacionamento existente entre as classes Centro e Universidade. Neste caso, o relacionamento está presente no mapeamento da classe Centro, através da anotação @ManyToOne. Exemplo de classe Java com mapeamento Hibernate, abaixo:

```

@Entity @Table(schema="anotacoes")
public class Centro {
    //...
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name="id_universidade",
                insertable=true, updatable=true)
    @Fetch(FetchMode.JOIN)
    @Cascade(CascadeType.SAVE_UPDATE)
    private Universidade universidade; //Métodos getters e setters
    //...
}
  
```

- **n-n (many-to-many):** Um relacionamento n-n implica em existir uma nova tabela para mapear o relacionamento no banco de dados. Denomina-se essa nova tabela como DEPARTAMENTO_CURSO, como mostrado na (FIGURA 13). Dessa forma, um departamento possui uma coleção de cursos e um curso uma

coleção de departamentos. A existência dessas coleções nas classes de domínio é opcional. Por exemplo, pode ser que em um sistema real não seja necessário saber todos os departamentos de determinado curso, mas se for realmente necessário, o *Hibernate* apresenta outros mecanismos para a obtenção desta informação.

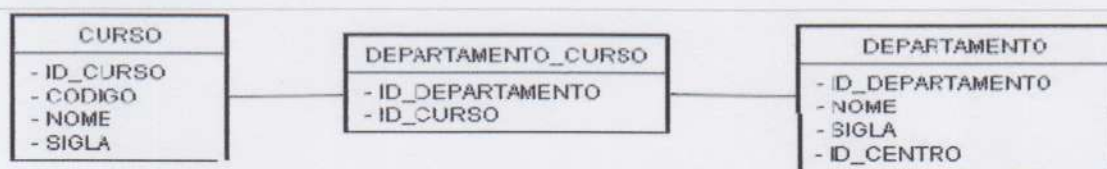


Figura 13: Relacionamento N-N de tabelas de um banco de dados.

Fonte: GALHARDO e LIMA, 2007.

2.8 PostgreSQL

Segundo a Comunidade Brasileira de *PostgreSQL*³, *ele* é um gerenciador de bancos de dados objeto-relacional, de código aberto. Possui suporte completo a chaves estrangeiras, junções, visões, gatilhos e procedimentos armazenados (em múltiplas linguagens). Inclui a maior parte dos tipos de dados do ISO SQL:1999, incluindo *INTEGER*, *NUMERIC*, *BOOLEAN*, *CHAR*, *VARCHAR*, *DATE*, *INTERVAL*, e *TIMESTAMP*. Suporta também o armazenamento de objetos binários, incluindo figuras, sons ou vídeos. Possui interfaces nativas de programação para C/C++, Java, *.Net*, *Perl*, *Python*, *Ruby*, ODBC, entre outros.

Como um banco de dados de nível corporativo, o *PostgreSQL* possui funcionalidades sofisticadas como o controle de concorrência multiversionado, recuperação em um ponto no tempo, *tablespaces*, replicação assíncrona, transações agrupadas (*savepoints*), cópias de segurança a quente (*online/hot backup*), um sofisticado planejador de consultas (otimizador) e registrador de transações sequencial para tolerância a falhas. O *PostgreSQL* suporta conjuntos de caracteres internacionais, codificação de caracteres *multibyte*, Unicode e sua ordenação por localização, sensibilidade a caixa (maiúsculas e minúsculas) e formatação. Possui alta escalabilidade, tanto na quantidade enorme de dados que pode gerenciar, quanto no número de usuários concorrentes que pode acomodar.

Com relação a arquitetura do *PostgreSQL* no servidor, Gonzaga (2007) descreve um processo *server* de segundo plano chamado *Postmaster* é iniciado com a finalidade de

³ <http://www.postgresql.org.br>

gerenciar os arquivos do banco de dados, aceitar conexões ao banco vindas das aplicações clientes e executar ações no banco de dados.

As aplicações clientes podem ser diversas, variando de uma ferramenta orientada a texto, até uma aplicação gráfica, ou um servidor *web* que acessa o banco para publicar páginas na *internet*.

Ainda segundo o autor, o *PostgreSQL* pode tratar múltiplas conexões concorrentes de clientes. Para isso, ele dispara um novo processo para cada conexão. A partir daí, o cliente e o novo processo servidor se comunicam sem intervenção do processo *post-master* original, que fica, então, aguardando novas conexões de clientes.

2.9 Apache Tomcat 6.0

Pertencente a Apache Software Foundation o tomcat 6.0 é um servidor de aplicações *web* para Java, ou um *container* de *servlets*.

Já segundo Zambon (2007) o *Tomcat* é essencialmente:

1. Um Servidor *Web*.
2. Uma aplicação que executa Java *Servlets*.
3. Uma aplicação que converte páginas JSP e documentos em Java *Servlets*.

Ainda segundo o autor, a arquitetura do Tomcat consiste numa série de componentes funcionais que podem ser combinados de acordo com regras pré-definidas. A estrutura de instalação de cada servidor é definido no arquivo *server.xml*, que está localizado no subdiretório *Conf*, do diretório de instalação do Tomcat.(FIGURA 14)

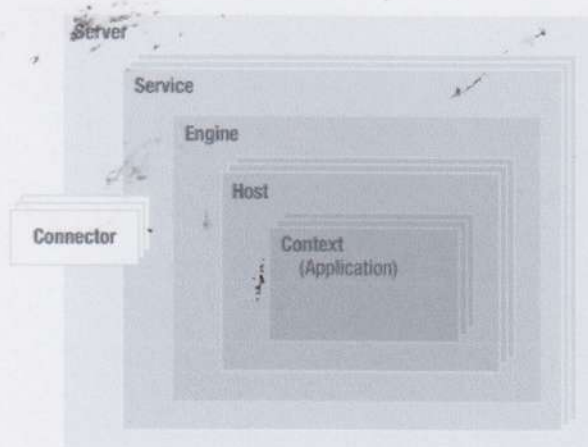


Figura 14: Arquitetura do Tomcat.

Fonte: ZAMBON, 2007.

- **Context:** É o principal elemento do grupo de componentes do *Tomcat* chamados *containers*, é representa uma única aplicação *web*. O *Tomcat* automaticamente instancia e configura o contexto padrão da aplicação, processa também as propriedades definidas no arquivo *web.xml*, do seu diretório de aplicação, e os torna disponíveis a aplicação.
- **Connector:** É o componente associado a porta TCP que gerencia a comunicação entre as aplicações e seus clientes. A configuração padrão do *Tomcat* inclui um gerenciador de comunicação HTTP, por padrão este conector espera por requisições vindas através da porta 8080. A presença do conector HTTP é o que torna o *Tomcat* um servidor *web*. As requisições de todas as aplicações vão através de uma única instância do conector, cada nova requisição gera uma instância de uma nova *Thread* dentro do conector, que se mantém viva enquanto durar a requisição.
- **Host:** O *container Host* representa uma hospedagem virtual, que realiza a associação entre um nome, como por exemplo *www.meuDominio.com.br*, e o servidor, um *Host* pode conter vários contextos (aplicações).

A configuração padrão do *Tomcat* inclui um host chamado *Localhost*, este *host* é encontrado pelo seu nome se ele estiver mapeado pelo DNS no IP do seu computador. O atributo do *container Host* chamado *appBase* define o diretório da aplicação com o *Tomcat* instalado.

- **Engine:** É o *container* de mais alto nível, pode conter vários *hosts*, mas não pode ser contido por outros componentes de *container*. A configuração padrão possui o *engine Catalina*, que contém o *host Localhost*, a *engine Catalina* gerencia todas as requisições vindas do conector HTTP, e envia de volta as respectivas mensagens. Enviando cada requisição para o *host* e *context* corretos, com base nas informações contidas em cada cabeçalho da requisição.
- **Service:** O componente *Service* tem como função um ou mais conectores a uma *engine* específica. Em sua configuração básica o *Service Catalina* já vem instalado, e realiza a associação entre os conectores HTTP a *engine* do *Catalina*.
- **Server:** É o mais alto componente, que representa a instância do *Tomcat*, pode conter um ou vários serviços, cada um com suas *engines* e conectores.

2.10 UML

A UML (Linguagem de Modelagem Unificada) caracteriza-se como uma linguagem de modelagem de terceira geração. A UML não diz o que você deve fazer, portanto ela não pode ser definida como uma metodologia de desenvolvimento, seu objetivo é auxiliar na visualização, no desenho e na comunicação entre objetos de um software ou sistema. A UML permite aos desenvolvedores visualizem os artefatos, caracterizado neste contexto como digramas, gerados pelo processo de desenvolvimento de software. A linguagem possui uma notação gráfica independente de qualquer tipo de processos que venha a ser utilizado.

De acordo com Seabra Júnior (2001) a UML é apenas parte de uma metodologia, uma linguagem de modelagem do sistemas, ela apenas disponibiliza as ferramentas necessárias para se criar e ler modelos, mas não aponta quais modelos deverão ser criados, nem quando deverão ser criados. Essa tarefa é de responsabilidade de um processo de desenvolvimento de sistemas.

Ainda de acordo com o autor, mesmo sendo independente de processo a UML necessitava interagir com uma metodologia específica que pudesse obter o máximo de proveito dos recursos da linguagem de modelagem unificada, então foi criado o processo unificado.

Com a integração do processo unificado à UML, repostas para quem está fazendo o que, quando e como puderam ser definidas e um padrão de desenvolvimento de sistemas orientados a objetos pode ser estabelecido.

2.12 Astah

O Astah *Professional UML* é uma ferramenta IDE para elaboração de artefatos da UML, realizando a integração entre as classes, os componentes, os diagramas de estado, casos de uso, etc. como mostrado no diagrama de classes do subsistema colaborativo do SIG Conimpe (APÊNDICE 1).

2.13 Eclipse Helios IDE

O Eclipse Helios é uma plataforma IDE focada no desenvolvimento de ferramentas e aplicações de software. Hoje o Eclipse é a IDE Java mais utilizada no mundo. Por possuir uma forte orientação ao desenvolvimento baseado em plug-ins e o amplo suporte ao

desenvolvedor com centenas de plug-ins que procuram atender as diferentes necessidades de diferentes programadores. A Eclipse Foundation⁴ mantém vários projetos envolvendo a sua IDE Eclipse, tornando-a extremamente adaptável às necessidades de cada cliente.

⁴ <http://www.eclipse.org/>

3 METODOLOGIA

3.1 Tipo de pesquisa

É na pesquisa que será utilizado diferentes instrumentos para se chegar a uma resposta mais precisa.

Este trabalho foi concebido através de um estudo de caso da Conimpe Engenharia.

Define-se estudo de caso como um tipo de pesquisa que se concentra no estudo de um caso particular, considerado representativo de um conjunto de casos análogos, por ele significativamente representativo. (SEVERINO, 2007)

A partir do aprofundamento do estudo de caso foram definidos os principais requisitos e regras de negócio para o desenvolvimento e customização do sistema de informação da Conimpe Engenharia, o SIG Conimpe.

3.1.1 SIG Conimpe

O SIG Conimpe (FIGURA 15) é um sistema desenvolvido para o gerenciamento de Recursos Humanos, Estoque e Finanças, além de ser um Sistema Colaborativo, sua finalidade é integrar todos esses setores da empresa e melhorar o gerenciamento da informação de forma global. (APÊNDICE 2)

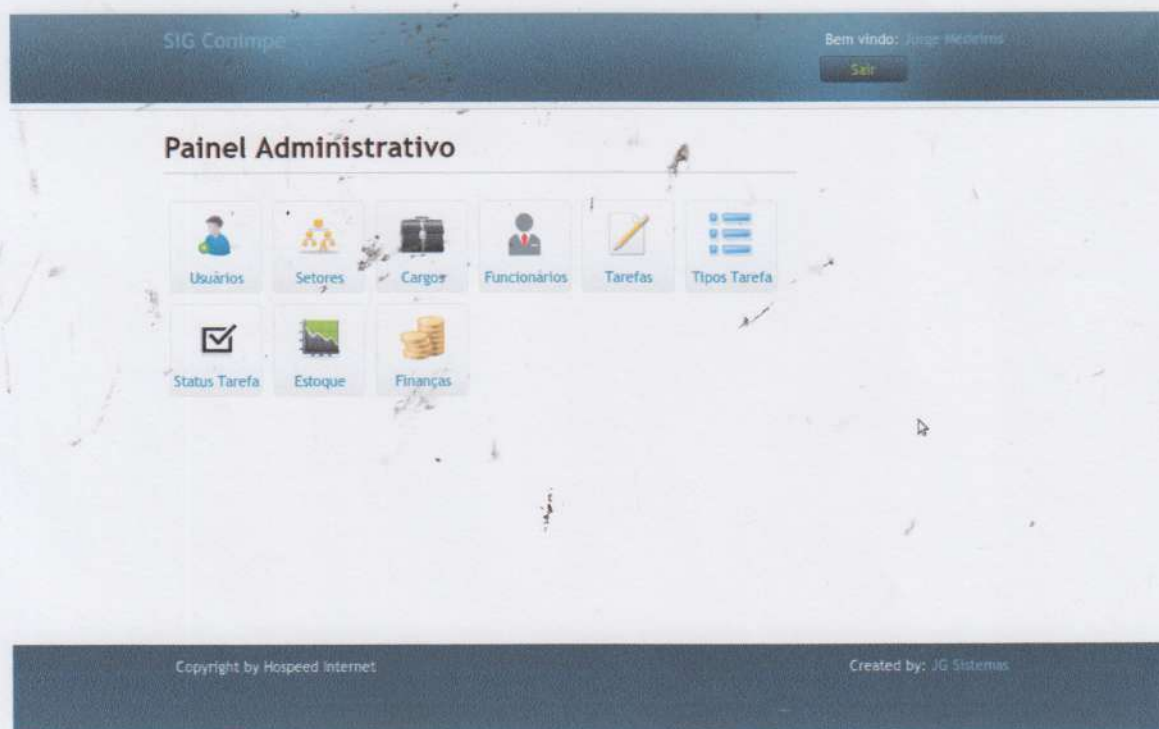


Figura 15: Tela inicial SIG Conimpe.

Fonte: O Autor.

Foram utilizados diversos *frameworks* para desenvolvimento do sistema. Para o mapeamento objeto-relacional, ou seja, a associação de classes *java* as tabelas do banco de dados PostgreSQL e gerenciamento das ações de consulta e persistência de dados foi utilizado o Hibernate Framework. Para a camada de visualização, onde é possível a interação entre os usuários e o sistema, foram integrados vários *frameworks*, o JSF ou JavaServerFaces, integrado ao JSF está o RichFaces, que além de fornecer suporte a requisições assíncronas (AJAX) ao servidor, também fornece componente gráfico de qualidade, como calendários e paginação de resultados. Outro *framework* utilizado na camada de visualização foi o *Facelets*, responsável pela utilização de *templates* de interface (Telas) e também de componentes de tela.

Para integrar todos esses *frameworks* e fazê-los trabalhar juntos de forma mais coesa foi utilizado o Jboss Seam Framework, que gerencia todas essas ferramentas e diminui a necessidade de uso de arquivos .xml, através de anotações e escopos próprios.

O SIG Conimpe segue a ideia de um sistema ERP, onde vários sistemas funcionais que operavam isoladamente, e às vezes até eram incompatíveis, são projetados para operar de forma integrada e dispondo essas informações de forma precisa e em tempo real aos gestores da empresa.

O sistema é composto por 4 (quatro) módulos:

- **Módulo de Recursos Humanos:** O módulo de recursos humanos é responsável pelo gerenciamento da força de trabalho da empresa. Suas principais funcionalidades são:
 - ✓ Gerenciamento de Funcionários (FIGURA 16)
 - ✓ Gerenciamento de Folha de Pagamento
 - ✓ Recrutamento de Força de Trabalho (Currículo on-line)
 - ✓ Gerenciamento dos Cursos obrigatórios para o desempenho das funções
 - ✓ Gerenciamento dos Exames Médicos funcionais
 - ✓ Controle de Frequência
 - ✓ Gerenciamento de Cargos (FIGURA 17)
 - ✓ Gerenciamento de Setores

SIG Conimpe

Bem vindo: Jorge Mendes

Sair

Gerenciamento de Funcionários

Campos Marcados Com * São de Preenchimento Obrigatório

Funcionários

Setor: *

--Selecione--

Cargo: *

--Selecione--

Foto: *

Nome: *

Email: *

Senha: *

Nível Acesso: *

--Selecione--

Salvar

Figura 16: Tela SIG Conimpe – Gerenciamento de Funcionários.
Fonte: O Autor.

SIG Conimpe

Bem vindo: Jorge Mendes

Sair

Gerenciamento de Cargos

Campos Marcados Com * São de Preenchimento Obrigatório

Cargos

Cargo: *

Salvar

Ord	Cargo	Ações
1	Analista de Sistemas	
3	AGG	
11	ASSESSOR	
13	Bolsista	
12	Consultor	

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

Copyright by Hospeed Internet

Created by: JG Sistemas

Figura 17: Tela SIG Conimpe – Gerenciamento de Cargos.
Fonte: O Autor.

- **Módulo de Estoque:** O módulo de controle de estoque visa prover informações em tempo real dos estoques de matéria prima da empresa, possibilitando tomadas de decisão mais eficientes pelos gestores. Suas principais funcionalidades são:
 - ✓ Gerenciamento de Material
 - ✓ Alerta de Níveis de Estoque
 - ✓ Gerenciamento de Fornecedores
 - ✓ Gerenciamento de Prestadores de Serviço
 - ✓ Gerenciamento de Patrimônio
 - ✓ Gerenciar Compras

- **Módulo de Finanças:** O módulo de finanças gerencia as funções mais básicas da contabilidade da empresa, sua função é prover aos gestores de forma clara as linhas gerais das finanças da instituição. Suas principais funcionalidades são:
 - ✓ Gerenciamento de Receitas
 - ✓ Gerenciamento de Despesas
 - ✓ Calendário de Contas

- **Módulo Sistema Colaborativo:** O módulo de Sistema colaborativo possibilita um maior fluxo e disponibilidade de informação dentro da empresa, é onde os gerentes poderão programar e controlar projetos e tarefas, bem como o desempenho dos colaboradores. No segmento operacional do sistema colaborativo existe uma integração com o módulo de Estoque, onde uma tarefa operacional automaticamente atualizará o banco de dados dos níveis de materiais. Suas principais funcionalidades são:
 - ✓ Gerenciamento de Projetos
 - ✓ Gerenciamento de Tarefas de Projetos
 - ✓ Ferramenta Wiki
 - ✓ Ferramenta de envio de mensagem coletiva
 - ✓ Gerenciamento de Tarefas Operacionais integrada ao controle de Estoque

4 CONCLUSÕES E RECOMENDAÇÕES

Com a implantação de um sistema ERP na Conimpe Engenharia ficou claro que o uso de um sistema que integre e padronize os setores e operações de uma empresa melhoram o gerenciamento de recursos humanos e materiais, e também aumenta sua produtividade, através de sistemas colaborativos e de controle de tarefas.

Em contrapartida também ficaram evidentes algumas dificuldades na concepção do projeto, a primeira de todas foi a barreira cultural em relação à tecnologia da informação, pois a maioria dos funcionários são técnicos, específicos em suas áreas de atuação profissional e não necessariamente administradores de empresa por formação, existe ainda um grande desconhecimento a respeito do que são, e o que fazem os softwares de gestão integrada nas pequenas empresas, e isso ocasiona uma desconfiança em relação a real necessidade de uma implementação de ERP, bem como sua eficácia, também ocasiona uma grande desvalorização do produto, tornando-o quase inviável comercialmente.

Outra barreira cultural foi mais específica em relação a mudança na metodologia de gestão, por serem práticas já difundidas na empresa, verificou-se uma maior dificuldade de aceitação e até mesmo de uso do novo sistema.

Diante destas dificuldades que foram levantadas durante a elaboração do Sistema Integrado de Gestão de Recursos para a Conimpe Engenharia, foi necessário um levantamento detalhado de cada um dos processos da empresa, de como são gerenciados, e de como interagem ou poderiam interagir entre si. Esse estudo prévio detalhado possibilitou uma customização onde o sistema se assemelha as antigas práticas, visando diminuir o tempo de treinamento e adaptação, bem como a rejeição dos usuários, mas sem perder de vista a qualidade e confiabilidade das informações.

Com a implantação em seu estágio final, aonde o uso do sistema irá se consolidar, novas necessidades de análise estão surgindo, como o desempenho do sistema, como adequar novas funcionalidades e melhorar processos já implementados, outro ponto que vem sendo analisado é o desenvolvimento de aplicativos para dispositivos móveis, essa possibilidade irá aumentar ainda mais geração de informação em tempo real na Conimpe Engenharia.

REFERÊNCIAS

ALLEN, Dan. **Seam in Action**. Greenwich (USA): Manning, 2008.

ARANDA, Bruno; WADIA, Zubin. **The Definitive Guide to Apache MyFaces and Facelets**. New York: Apress, 2008.

CAIÇARA JUNIOR, Cícero. **Sistemas Integrados de Gestão ERP: uma abordagem gerencial**. 3ª Edição. Curitiba: Editora IBPEX, 2008.

CARMO, Raquel do; FREIRE, Rafael. **JBOSS SEAM: Framework para Java EE5**. (2009) Disponível em <http://www.slideshare.net/raquelcarsi/jboss-seam-presentation>. Acesso em Outubro/2011.

DEITEL, H.M.; DEITEL, P. J. **JAVA Como Programar**. 3ª Edição. Porto Alegre: Bookman, 2001.

ELLIOTT, James; O'BRIEN, Timothy M.; FOWLER, Ryan. **Dominando Hibernate**. Rio de Janeiro: AltaBooks, 2009.

FARLEY, Jim. **Practical JBoss Seam Projects**. Apress: New York, 2007.

FONSECA, Daniel Berti. **Integração de Sistemas Corporativos Complexos com Java EE**. Monografia (Bacharelado em Ciência da Computação) – Curso de Ciência da Computação da Faculdade de Jaguariúna, Jaguariúna, 2006.

GALHARDO, Raphaela; LIMA, Gleydson de A. Ferreira. **Hibernate com Anotações**. (2007). Disponível em <http://pt.scribd.com/doc/27724352/9/SessionFactory-org-hibernate-SessionFactory>. Acesso em Novembro, 2011.

GEARY, David; HORSTMANN, Cay. **Core JavaServer Faces**. Segunda Edição. Rio de Janeiro: AltaBooks, 2007.

GONZAGA, Jorge Luiz. **Dominando o PostgreSQL**. Rio de Janeiro: Editora Ciência Moderna, 2007.

KATZ, Max. **Practical RichFaces**. Apress: New York, 2008.

O'BRIEN, James A. **Sistemas de Informação e as decisões gerenciais na era da internet**. São Paulo: Saraiva, 2001.

PAIVA, Jonas; MARTINS, Roberto; ANDRADE, Yuri. **Facelets**. Disponível em www.riopomba.ifsudestemg.edu.br/%2Fdcc%2Fdcc%2Fmateriais%2F665174660_Facelets.pptx&ei=3ELdTrTHGc7wgge-x_jXAg&usg=AFQjCNFHZ-uDNX7jXW1LNKCEaP2u3jkYsQ&cad=rja Acesso em Novembro/2011.

SEABRA JÚNIOR, Rodolfo Moacir. **Análise e Projeto Orientado a Objetos usando UML e o Processo Unificado**. Centro de Ciências Exatas e Naturais. Universidade Federal do Pará. Campus Universitário do Guamá. Belém: UFPA, 2001.

SEVERINO, Antonio Joaquim. **Metodologia do Trabalho Científico**. 3ª Edição Revista e Atualizada. 3ª Reimpressão. São Paulo: Cortez Editora, 2007.

SOUZA, Rafael de Paula. **Conhecendo o RichFaces – Parte I**. Disponível em: <http://www.devmedia.com.br/post-6440-Conhecendo-o-RichFaces-Parte-I.html> Acesso em Outubro/2011.

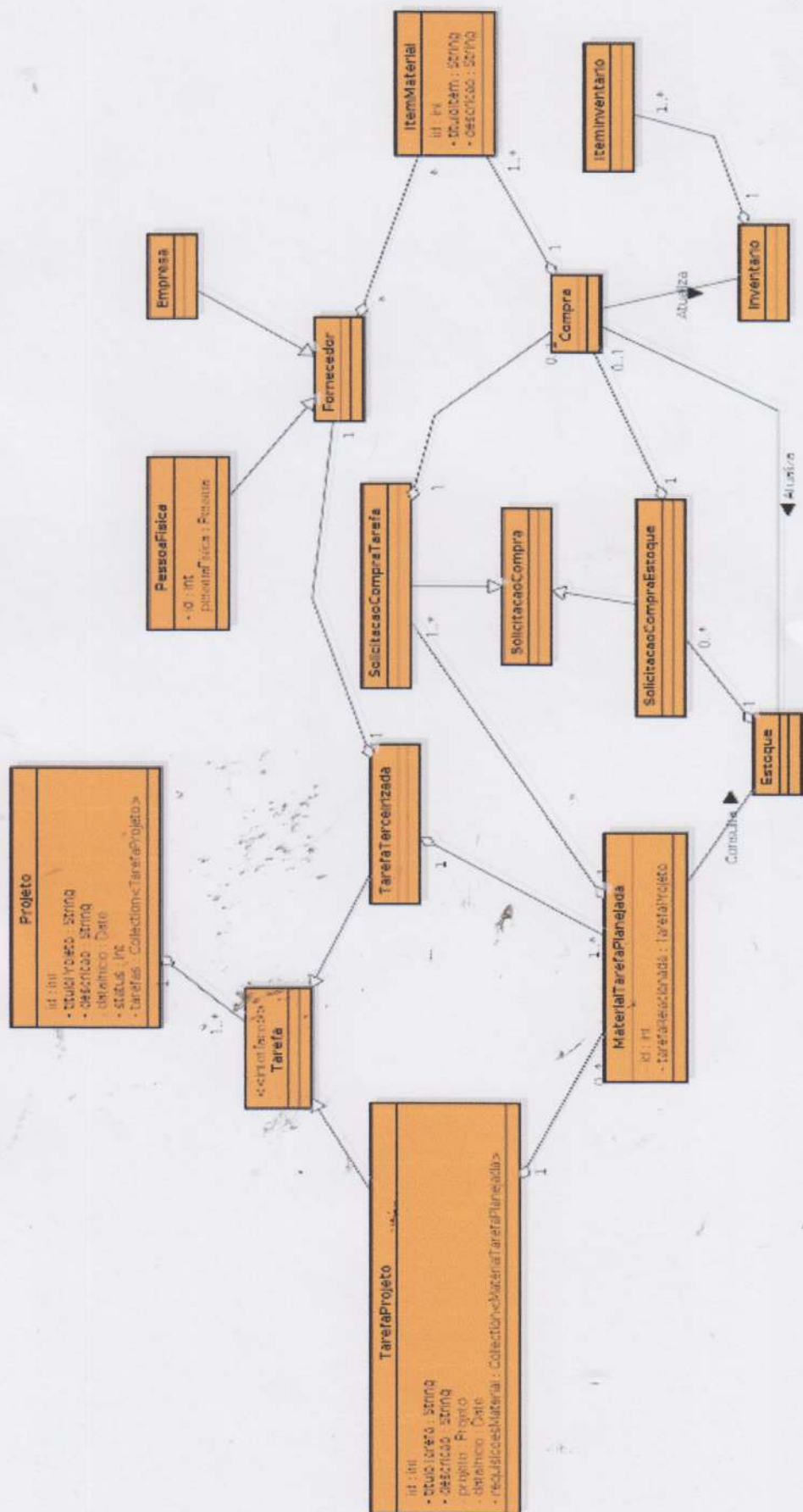
TACLA, Cesar A. **Tecnologia de Orientação a Objetos** (2007). Disponível em: <http://www.dainf.ct.utfpr.edu.br/~tacla/Tecno-OO>. Acesso em Novembro/2011.

VIEIRA, Robson Paz. **Globalização, avanço tecnológico e a necessidade** (2007). Disponível em <http://www.gestaodecarreira.com.br/coaching/criatividade-inovacao/globalizacao-avanco-tecnologico-e-a-necessidade.html>. Acesso em Outubro/2011.

ZAMBON, Giulio. **Beginning JSP, JSF and Tomcat Web Development**. Apress: New York, 2007.

APÊNDICE

APÊNDICE 1 - Diagrama de Classes do Subsistema colaborativo do SIG Conimpe.



Fonte: O Autor.
Software: Astah

