

Universidade Estadual de Maringá  
Centro de Tecnologia - Departamento de Informática  
Especialização em Desenvolvimento de Sistemas para *Web*

## **ESTUDO DO DESENVOLVIMENTO DE UMA APLICAÇÃO WEB UTILIZANDO O SPRING FRAMEWORK**

ANA CAROLINA CARVALHO DE PAULA SOUZA

Prof. Msc. Munif Gebara Junior  
**Orientador**

Maringá  
2011

Universidade Estadual de Maringá  
Centro de Tecnologia - Departamento de Informática  
Especialização em Desenvolvimento de Sistemas para *Web*

## **ESTUDO DO DESENVOLVIMENTO DE UMA APLICAÇÃO WEB UTILZANDO O SPRING FRAMEWORK**

ANA CAROLINA CARVALHO DE PAULA SOUZA

**Trabalho submetido à Universidade Estadual de Maringá  
como requisito para obtenção do título de Especialista  
de Desenvolvimento de Sistemas para *Web*.**

Orientador: Prof. Msc. Munif Gebara Junior

Maringá  
2011

Universidade Estadual de Maringá  
Centro de Tecnologia - Departamento de Informática  
Especialização em Desenvolvimento de Sistemas para *Web*

## **ESTUDO DO DESENVOLVIMENTO DE UMA APLICAÇÃO WEB UTILZANDO O SPRING FRAMEWORK**

ANA CAROLINA CARVALHO DE PAULA SOUZA

**Trabalho submetido à Universidade Estadual de Maringá  
como requisito para obtenção do título de Especialista  
de Desenvolvimento de Sistemas para *Web*.**

Orientador: Prof. Msc. Munif Gebara Junior

Maringá  
2011

Maringá, 4 de abril de 2011.

---

Prof. Msc. Flávio Rogério Uber

---

Prof. Msc. Flávio Rogério Uber

---

Prof. Msc. Munif Gebara Junior (orientador)

DEDICATÓRIA

Ao meu marido Reginaldo .

## AGRADECIMENTOS

Agradeço primeiramente a **DEUS**, pela força de vontade e superação que tive durante esse trabalho.

Ao meu pai **João Antonio de Paula**, que mesmo longe sempre me incentivou a continuar estudando.

A minha sogra Tereza Ferreira de Souza por me apoiar como se fosse a minha própria mãe.

Aos meus filhos por agüentar ficar sem atenção da mãe durante o desenvolvimento do trabalho.

Ao meu orientador, **Munif Gebara Junior**, que me instruiu, corrigindo erros, e lendo esse trabalho Agradeço a ele por toda sua paciência,dedicação, conselhos.

## RESUMO

A Internet vem crescendo, e a busca de informações a todo momento em qualquer lugar está se tornando uma necessidade. Os sistemas de informação das empresas não podem mais ficar limitados a intranet onde, os diretores, os gerentes e os vendedores precisam estar conectados para tomar decisões imediatas. Com isso o desenvolvimento de software para web vem crescendo, e a necessidade de profissionais capacitados e novas tecnologias para agilizar o processo de desenvolvimento também. Analisando as novas tecnologias de informação o framework Spring é uma alternativa que oferece soluções para desenvolvimento de aplicações web. Esse trabalho apresenta uma análise do desenvolvimento de uma aplicação com o framework Spring, e realizado uma análise de produtividade no desenvolvimento e na utilização da aplicação.

**Palavras Chaves** : Java, Framework, Spring e Sistemas WEB

## **ABSTRACT**

The Internet is growing, and the search for information at any time and anywhere is becoming a necessity. The information systems of companies can no longer be limited from intranet, directors, managers and salespeople need to be connected to take immediate decisions. With this software development for Web is growing, and the need for trained professionals and new technologies to streamline the development process as well. Analyzing the new information technologies Spring framework is an alternative that offers solutions for web application development. This paper presents an analysis of the development of an application with the Spring framework, and conducted an analysis of productivity in the development and application usage.

## LISTA DE FIGURAS

Figura 1. Separação de Interesses com POA .....	19
Figura 2. Objeto Json.....	21
Figura 3. Array Json.....	22
Figura 4.: Arquitetura interna do hibernate.....	22
Figura 5. Módulos do Spring Framework. ....	25
Figura 6.Principais módulos do Spring.....	28
Figura 7. Spring Portifólio .....	30
Figura 8. O ciclo de vida de uma requisição Spring MVC .....	33
Figura 9. Modelo de caso de uso Manter Cotação. ....	35
Figura 10. Estrutura do Dinamic Web Project .....	36
Figura 11. Pasta lib .....	37
Figura 12. Diagrama de Classes do Projeto.....	38
Figura 13. Classe Produto .....	39
Figura 14. ContactDAO.....	41
Figura 15. Interface IContactDAO. ....	41
Figura 16. DBconfig.xml.....	42
Figura 17 . Classe Produto Service.....	43
Figura 18. Produto Controller.....	44
Figura 19. Tela inicial do Cotação Express.....	45
Figura 20. Parte do código Java Script para montagem da tela cadastro de Produto. ....	46
Figura 21. Tela de Cadastro de Contato .....	47
Figura 22. Tela de Cadastro de Produto .....	48
Figura 23. Tela de cadastro da Cotação. ....	49
Figura 24. Validação Produto na cotação. ....	49

## ABREVIATÓES

HTML *HyperText Markup Language*

JSP *Java Server Pages*

CSS *Cascading Style Sheets*

J2EE *Java 2 Enterprise Edition*

EJB *Enterprise Java Beans*

AOP *Aspect-Oriented Programming*

IOC *Inversion of Control*

DI *Dependence Injection*

ORM *Object-relational mapping*

JPA *Java Persistence API*

JDBC *Java Database Connectivity*

JTA *Java Transaction API*

JDO *Java Data Objects*

JMS *Java Message Service*

JMX *Java Management Extensions*

MVC *Model-view-controller*

RMI *Remote Method Invocation*

IIOP *Internet Inter-ORB Protocol*

POJO *Plain Old Java Objects*

XML *Extensible Markup Language*

JAAS *Just Another Scripting Syntax)*

LDAP *Lightweight Directory Access Protocol*

DAO *Data Access Object*

WSDL *Web Services Description Language*

AJAX *Asynchronous Javascript And XML*

API *Application Programming Interface*

DOM *Document Object Model*

## SUMÁRIO

<b>1.INTRODUÇÃO .....</b>	<b>12</b>
1.1 ORGANIZAÇÃO DO TRABALHO .....	13
<b>2. REVISÃO BIBLIOGRÁFICA .....</b>	<b>14</b>
2.1 INTERNET .....	14
2.2 SPRING.....	14
2.4 PROGRAMAÇÃO ORIENTADA A ASPECTOS.....	17
2.5 EXTJS .....	19
2.6 JSON.....	20
<b>3. FRAMEWORK SPRING.....</b>	<b>25</b>
3.2 SPRING MVC.....	31
<b>4. ESTUDO DE CASO .....</b>	<b>35</b>
4.1 PRÉ-REQUISITOS.....	36
<b>5. RESULTADOS.....</b>	<b>47</b>
<b>6. CONCLUSÃO .....</b>	<b>51</b>
<b>7. REFERÊNCIAS.....</b>	<b>53</b>

## 1.INTRODUÇÃO

A Internet vêm crescendo , e a necessidade de ter as informações a todo o momento em qualquer lugar está se tornando uma necessidade. O sistemas de informação das empresas não podem mais permanecer limitados a intranet da onde , os diretores, gerentes precisam saber o que acontece na organização a qualquer momento. Com isso o desenvolvimento de software para web vem crescendo, e necessidade de profissionais capacitados e novas tecnologias para agilizar esse processo de desenvolvimento também.

Analisando as novas tecnologias de informação o framework Spring é uma alternativa que oferece soluções para desenvolvimento de aplicações web.

Cotação Express é o resultado de um estudo de caso de desenvolvimento de uma aplicação on-line de pequeno porte utilizando o framework Spring.

O estudo demonstra que o Spring framework é um framework de aplicação completa podendo utilizar integração com outros frameworks de aplicação especifica como o Hibernate para persistência de dados.

O estudo de caso realizado no Centro Universitário de Maringá (CESUMAR) demonstra que e os testes realizados com os atendentes da clinica de estética o sistemas de cotação Express agiliza no atendimento aos clientes por telefone e pessoalmente.

O funcionamento do cotação Express basicamente o atendente cadastra um contato e seleciona a opção de cotação. Na tela de cotação Seleciona o nome do contato e o produto que deseja consultar o valor e o cotação express mostra resultado em tela e possui a opção grava a cotação se desejar que futuramente vire um atendimento na clinica.

Esse projeto visa avaliar o desenvolvimento de uma aplicação utilizando o framework Spring e mostrar as possibilidades futuras para desenvolvimento de uma aplicação complexa utilizando a estrutura do framework.

Foi realizada uma simulação de velocidade de execução da aplicação em relação ao processo utilizado na clinica de estética manual.

## 1.1 ORGANIZAÇÃO DO TRABALHO

No Capítulo 2 são apresentados os conceitos necessários para o entendimento deste trabalho: internet , java, inversão de controle e programação orientada a aspectos.O objetivo é mostrar o panorama atual dos assuntos abordados e contextualizar a abordagem proposta neste trabalho com o processo de desenvolvimento de software utilizando frameworks.

O Capítulo 3 define o framework Spring, os módulos do Spring framework.

O Capítulo 4 trata do estudo de caso realizado no Centro Universitário de Maringá,na clinica de estética.

No Capitulo 5 mostra o resultado do trabalho o layout das telas, algumas partes de códigos.

Por fim, o Capítulo 6 apresentará as conclusões obtidas com o desenvolvimento deste trabalho e as propostas de alguns trabalhos futuros que poderão ser desenvolvidos.

## 2. REVISÃO BIBLIOGRÁFICA

Neste capítulo apresenta-se um breve estudo de algumas ferramentas relacionadas ao Spring Framework contextualizando a tecnologia utilizada.

### 2.1 INTERNET

A internet uma rede em escala mundial de milhões de computadores interligados . Segundo DEITEL ,2005 *“foi desenvolvida há quase quatro décadas com o financiamento fornecido pelo U.S. Department of defense (Departamento de Defesa dos Estados Unidos).”* Projetada inicialmente para conectar os principais sistemas de computadores de universidades e organizações de pesquisa. Atualmente a internet é acessível a centenas de milhões de computadores no mundo.Com o surgimento da World Wide Web os usuários de computador podem localizar e visualizar documentos baseados em multimídia sobre quase qualquer assunto pela internet – a Internet explodiu, tornando-se um dos principais mecanismos de comunicação do mundo.

Os aplicativos de hoje estão sendo escritos para se comunicarem utilizando a internet. Segundo Deitel , 2005.” *A Internet une computação e tecnologias de comunicação.”* Com o uso da internet torna o trabalho mais fácil e as informações acessíveis mundialmente de maneira instantânea e conveniente. Aplicativos desenvolvidos em Java podem ser escritos para executar na internet em qualquer lugar do mundo, reduzindo significativamente o tempo e o custo de desenvolvimento de sistemas para as corporações.

### 2.2 SPRING

Java foi criada por um grupo de Engenheiros da Sun, liderados por Patrick Naughton e James Gosling, com objetivos de criar uma linguagem pequena que

pudesse ser utilizada em dispositivos de consumidores, como swichboxes de TV a Cabo com pouco memória ou potência, e ainda gerar um código robusto.

Os requisitos de código pequeno, robusto e independente a plataforma levaram a plataforma Java a ser utilizada universalmente, todo tipo de aplicação pode ser desenvolvida nela. J2EE foi criada para desenvolvimento de aplicações de grande porte deixando muito complexa o desenvolvimento de aplicações menores. Para amenizar e reduzir esse problema surgiram os frameworks, criados pela comunidade de desenvolvedores Java.

Segundo WALL , 2008

Spring é um framework de código aberto, criado por Rod Johnson e descrito em seu livro, *Expert One-on-One: J2EE Design e Development*. Foi criado para lidar com a complexidade de desenvolvimento de aplicativos corporativos. O Spring torna possível usar simples JavaBeans para conseguir coisas que antes só eram possíveis com EJBs. Porém, a utilidade do Spring não é limitada ao desenvolvimento do lado do servidor. Qualquer aplicativo em Java pode se beneficiar do Spring em termos de simplicidade, testabilidade e baixo acoplamento.

Ao fazer isso adota uma filosofia simples que J2EE deve ser fácil de usar. O Spring foi projetado para que um bom design é mais importante que a tecnologia subjacente. Java Beans fracamente acoplados através de interfaces é um bom modelo código deve ser fácil de testar.

Recentemente, o mundo Java testemunhou uma dramática mudança para fora da assim chamada arquitetura “pesada”, como o Enterprise Java Beans (EJB) em direção a estruturas mais leves, como o Spring. Os Serviços complexos como sistemas de gerenciamento de transações, foram substituídos por alternativas simples, como o Hibernate e a programação orientada por aspectos (AOP). Segundo VUCOKITI, 2008 , *“O Spring fornece um container abrangente e leve, baseado no princípio da Inversão de Controle (IoC, na sigla em inglês)”*, no qual, podem ser desenvolvidas novas aplicações baseadas no contêiner do Spring. Segundo VUCOKITI, 2008 *“O Spring fornece serviços úteis, trazendo junto projetos de código aberto altamente competentes, numa estrutura coesa e única.”*

O núcleo da Estrutura Spring é baseado no princípio da Inversão de Controle (IoC). Aplicações que seguem este princípio usam configurações que descrevem as dependências entre seus componentes. Depois, fica a cargo da estrutura IoC satisfazer as dependências configuradas. A “inversão” significa que a aplicação não controla sua estrutura; é responsabilidade da estrutura IoC fazê-lo.

Segundo VUCOKITI , 2008:

A implementação da DI (Injeção de dependência ) põe o foco o pouco acoplamento : os componentes de sua aplicação devem presumir o mínimo possível dos outros componentes. A maneira mais fácil de se obter o baixo acoplamento no Java é codificar para interfaces.

O conceito importante é que cada componente não saberá qual implementação concreta ele estará usando; ele só verá uma interface.

As interfaces e a DI são mutuamente benéficas. O desenho e a codificação de uma aplicação por interfaces possibilitam uma aplicação flexível, que é muito mais conveniente para testes de unidade. Usando-se a DI, você reduz a quantidade de código extra de que precisa, num desenho baseado em interfaces, a quase zero. Da mesma forma, usando interfaces, você poderá obter o máximo da DI, porque seus beans poderão utilizar qualquer implementação da interface para satisfazer suas dependências.

A utilização da DI oferece benefícios para redução de código de colagem sendo um dos maiores pontos positivos da DI é sua habilidade de reduzir, dramaticamente, a quantidade de código que você tem de escrever para colar os diferentes componentes de sua aplicação.

A exteriorização de dependências é outro benefício que a DI oferece podendo exteriorizar a configuração de dependências, o que lhe permite reconfigurar facilmente, sem a necessidade de recompilar a aplicação.

*Gerenciamento de dependência em um único lugar:* Numa abordagem tradicional do gerenciamento de dependências, você cria instâncias de suas dependências onde elas são necessárias — dentro da classe dependente. Pior ainda, em típicas aplicações grandes, você normalmente usa um produtor ou localizador para encontrar os componentes dependentes. Isto significa que seu

código depende do produtor ou localizador, tanto quanto da dependência real. Com exceção das mais triviais das aplicações, você terá dependências espalhadas pelas classes em sua aplicação, e alterá-las pode ser problemático. Quando você usa a DI, toda a informação sobre as dependências é de responsabilidade de um único componente tornando o gerenciamento de dependências muito mais simples e menos inclinado a erros.

*Melhoramento da possibilidade de teste:* Quando desenha suas classes para a DI, você torna possível substituir facilmente as dependências.

*Melhorar designer de aplicações:* Desenhar para DI significa, em geral, desenhar por interfaces. Uma típica aplicação orientada por injeção é desenhada de forma que todos os componentes principais sejam definidos como interfaces, e, depois, implementações concretas destas interfaces são criadas e conectadas em conjunto, usando-se o contêiner da DI.

## 2.4 PROGRAMAÇÃO ORIENTADA A ASPECTOS

A programação orientada a aspectos permite separar e organizar o código de acordo com sua importância para a aplicação. De acordo com (GOETTEN,2006) *“A orientação a Aspectos (AO) é um paradigma que estende a Orientação a Objetos (e outros, como o paradigma estruturado) introduzindo novas abstrações.”*

A orientação a objetos está consolidada porém não conseguiu cumprir todas as promessas, novos elementos entre eles a orientação a aspectos foram criados para suprir deficiências na capacidade de representação de algumas situações que a orientação a objetos se propunha a cumprir e não foram atendidas.

Um dos elementos centrais da OA é o conceito de Interesse, que visa organizar as características relevantes da aplicação.

Segundo (GOETTEN ,2006) *“Um interesse pode ser dividido em uma série de aspectos que representam os requisitos. “Os aspectos podem ser agrupados no domínio da aplicação, compondo os interesses funcionais, que formam a lógica de negócio.*

Cada aspecto encapsula uma funcionalidade que atravessa outras classes da aplicação. Um aspecto é combinado com as classes que ele afeta de acordo com as definições dentro do aspecto. Um aspecto pode introduzir métodos, atributos e interfaces.

Segundo (GOETTEN, 2006)

A implementação de vários interesses sistêmicos e funcionais em um mesmo módulo com a orientação a objeto resulta no código chamado de emaranhado (Tangled Code).O código espalhado e emaranhado dificulta da manutenção e a reutilização de código.

De acordo com (GOETTEN,2006)

A programação orientada a aspectos foi criada no fim da década de 1990, mais precisamente no ano de 1997, em Palo Alto, nos laboratórios da Xerox, por Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, JeanMarc Loingtier e John Irwin.

A POA estende outras técnicas, como a POO ou programação estruturada, propondo não apenas uma decomposição funcional, mas também sistêmica do problema. Permitindo que a implementação do sistema seja separada em requisitos funcionais e não funcionais, disponibilizando a abstração de aspectos para a decomposição de interesses sistêmicos.

Segundo (GOETTEN,2006) *“O principal objetivo da POA consiste em separar o código referente ao negócio do sistema dos interesses transversais, de uma forma bem definida e centralizada.”* A separação e a centralização propiciada pela POA são indicadas pela Figura 1.

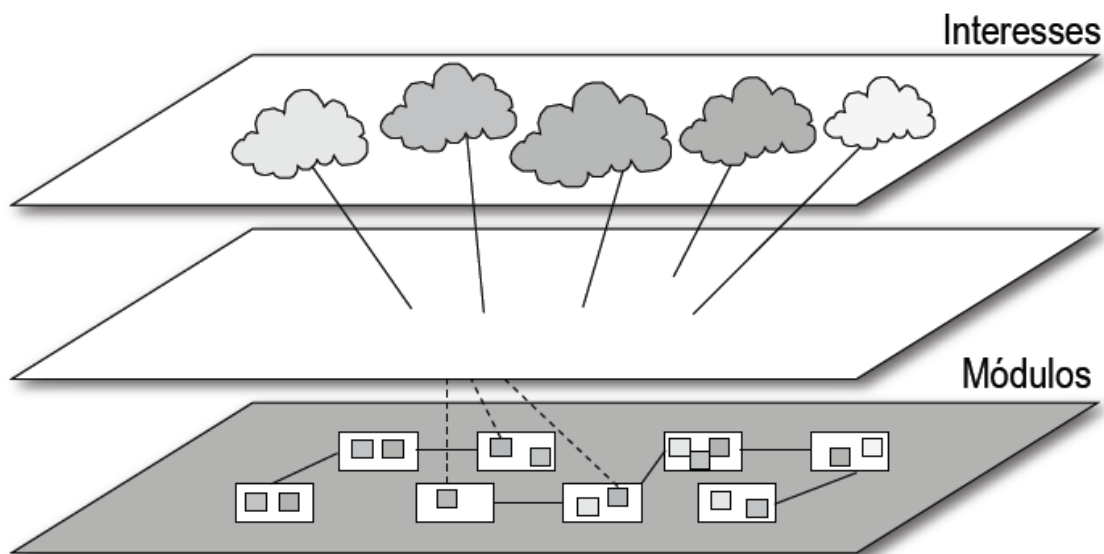


Figura 1. Separação de Interesses com POA

Fonte: GOETTEN (2006)

A programação orientada pelo aspecto (AOP, na sigla em inglês) ou POA é uma das tecnologias do momento, no espaço da programação. A POA lhe permite implementar a lógica entrelaçada — ou seja, lógica que se aplica em muitas partes de sua aplicação — num único lugar e, depois, ter esta lógica automaticamente aplicada em toda a aplicação. A POA está no centro das atenções, no momento; entretanto, por trás de toda a publicidade, está uma tecnologia verdadeiramente útil, que tem lugar em qualquer caixa de ferramentas de um desenvolvedor Java.

## 2.5 EXTJS

A Biblioteca Ext JS 3.0 segundo (Garcia , 2010) “*é um poderoso framework de UI para a construção de aplicações ricas e robustas, originalmente desenvolvido por Jack Slocum em 2006.*”

Desde então, Ext JS tem sofrido um crescimento explosivo, porque aborda a necessidade dos desenvolvedores web para ter um framework para desenvolvimento da interface do usuário, com componentes e modelos de eventos. O Ext JS torna ainda único no espaço na crescente concorrência da Web 2.0 de bibliotecas.

Ext JS fornece um rico conjunto de utilitários DOM(Document Object Model) e widgets. Embora possamos ficar animados com o que se vê na página de exemplos, é o que está na camada de baixo que é mais emocionante.

Ext JS vem com uma suíte completa de ferramentas de gerenciamento de layout para lhe dar total controle sobre a organização e manipulando a interface do usuário conforme a necessidade.

Uma camada de baixo existe o que é conhecido como o modelo de componente e contêiner modelo, cada uma desempenhando um papel importante na gestão como as interfaces são construídas.

Quase todos os elementos da interface do usuário no framework é altamente customizável, permitindo-lhe a opção de habilitar e desabilitar recursos, funções de substituição, e usar extensões personalizadas e plug-ins. Ext JS pode ser usado para criar uma aplicação Web de página inteira.

Componente Container e modelos do componente Container e modelos desempenham um papel fundamental na gestão de interfaces com o Ext JS e são parte da razão Ext JS destaca-se o resto das bibliotecas Ajax e estruturas.

## 2.6 JSON

JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é um formato leve de troca de dados. De acordo com a documentação “*JSON é em formato texto e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Perl, Python e muitas outras.*” Estas características fazem com que JSON seja um formato ideal para troca de dados.

De acordo com a documentação:

JSON está constituído em duas estruturas:

Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um object, record, struct, dicionário, hash table,

keyed list, ou arrays associativas. Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma array, vetor, lista ou sequência.

O Spring possui suporte ao formato JSON integrando ao módulo Spring MVC utilizando controladores padrão.

Segundo a documentação :*“Um objeto JSON é um conjunto desordenado de pares nome/valor. Um objeto começa com { (chave de abertura) e termina com } (chave de fechamento). Cada nome é seguido por : (dois pontos) e os pares nome/valor são seguidos por , (vírgula).”* Conforme demonstra a figura 2 extraída do site [www.json.org](http://www.json.org).

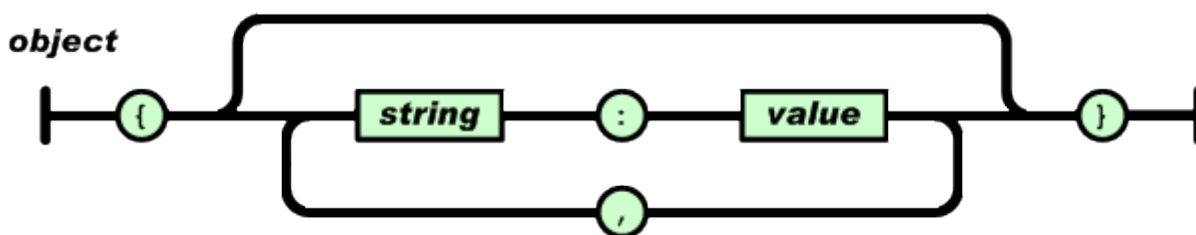


Figura 2. Objeto Json

Fonte : Json Reference (2010)

*“Uma array é uma coleção de valores ordenados. O array começa com [ (conchete de abertura) e termina com ] (conchete de fechamento). Os valores são separados por , (vírgula).”* Conforme demonstra a figura 3 extraída do site [www.json.org](http://www.json.org).

Um valor (value, na Figura 3) pode ser uma cadeia de caracteres (string), ou um número, ou true ou false, ou null, ou um objeto ou uma array. Estas estruturas podem estar aninhadas.

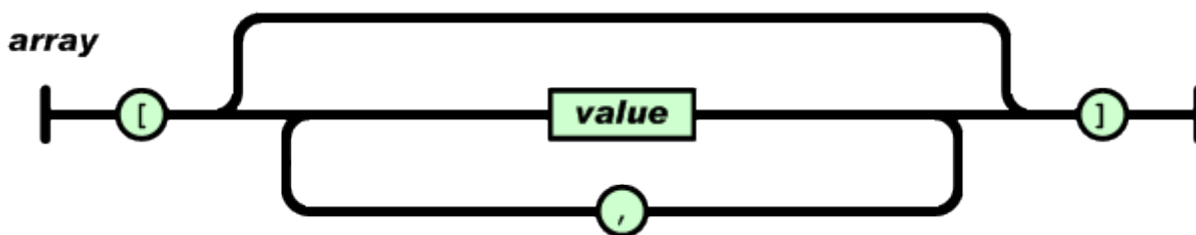


Figura 3. Array Json.

Fonte : Json Reference (2010)

## 2.7 HIBERNATE

Bauer e King (2005) explicam que o “*Hibernate é um framework de persistência que tem como finalidade armazenar objetos Java em bases de dados relacionais ou fornecer uma visão orientada a objetos de dados relacionais existentes.*” Isso se dá porque o *framework* utiliza arquivos de configuração *XML* para fazer o mapeamento dos dados contidos nas colunas de uma tabela em uma base de dados relacional para os atributos de uma classe *Java*.

A Figura 4 ilustra o funcionamento do *framework* no mapeamento dos objetos da aplicação para as tabelas de uma base de dados.

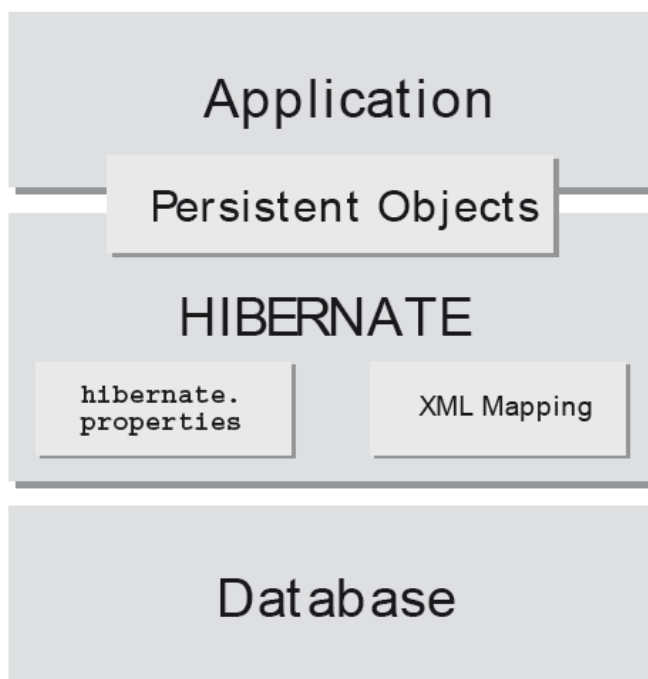


Figura 4.: Arquitetura interna do hibernate

O objeto persistente é passado para o *hibernate* que valida sua estrutura de dados através do arquivo *XML* de mapeamento de dados, se a estrutura de dados do objeto e da tabela na base de dados estiver compatível, o *hibernate* inicia uma transação na base de dados e persiste as informações.

Em uma aplicação orientada a objetos, o *hibernate* permite que um objeto criado por uma aplicação seja armazenado em bancos de dados relacionais, conservando o estado do objeto, podendo buscar esse mesmo objeto futuramente com o mesmo estado. O *hibernate* não se limita a buscar os objetos e todas as suas referências de uma só vez, após o objeto e suas referências terem sido salvos no banco de dados, pode-se buscar apenas o objeto desejado e caso seja necessário é feita a busca por suas referências.

Uma aplicação orientada a objetos não trabalha diretamente com a representação tabular das entidades de negócio como mostram (Bauer e King 2005); a aplicação tem seus próprios modelos das entidades de negócio. Se a base de dados tiver tabelas para artigo e para oferta, define-se para a aplicação as classes Artigo e Oferta, evitando de se trabalhar diretamente com os registros retornados por uma instrução SQL para depois ajustá-la a lógica de negócio do modelo da aplicação, e utiliza-se apenas os objetos das classes Artigo e Oferta para realizar as operações dentro do sistema. Essa abstração do conceito de tabelas de banco de dados segundo (Bauer e King 2005) "torna o desenvolvimento da camada de persistência de aplicações orientadas a objetos mais simples e intuitiva."

O Mapeamento Objeto Relacional é a tecnologia que provê a persistência de forma automatizada e transparente dos objetos em uma aplicação em tabelas de uma base de dados relacional, usando metadados para fazer a troca de dados entre os objetos e a base de dados, (BAUER e KING, 2005) fala que "o mapeamento objeto-relacional essencialmente trabalha com dados transformados de uma representação para a outra, isto implica em determinadas perdas no desempenho".

Entretanto, se o mapeamento objeto-relacional for executado como um *middleware*, há muitos ganhos e otimizações que não existiriam para uma camada codificada sem esse tipo de persistência.

Como mostra STONEBRAKER (1996) mapeamento objeto-relacional foi criado com a proposta de suprir as carências do modelo relacional, oferecendo toda a naturalidade para modelar objetos complexos e suas características peculiares que os bancos de dados orientados a objetos propunham, e ao mesmo tempo ter

toda carga tecnológica desenvolvida para os bancos relacionais em anos de desenvolvimento. A proposta é oferecer uma interface orientada a objetos e de forma transparente para esse, modelar tudo usando o paradigma que melhor se adequar.

### 3. FRAMEWORK SPRING

O Framework Spring é composto de vários módulos bem definidos conforme mostra a figura 3. Segundo (WALLS, 2008) “Ao serem tomados como um todo, estes módulos fornecem tudo o que é necessário para desenvolver aplicativos para soluções corporativas.” Mas não é necessário ter o Framework Spring como base para desenvolver aplicações.

Os módulos do Spring são livres podendo ser escolhidos o que melhor se adaptarem ao aplicativo em desenvolvimento. De acordo com WALLS , 2008 “O Spring oferece pontos de integração com outros frameworks e bibliotecas, não tendo que desenvolver por si próprio.” Os módulos do Spring são demonstrados na figura 5.

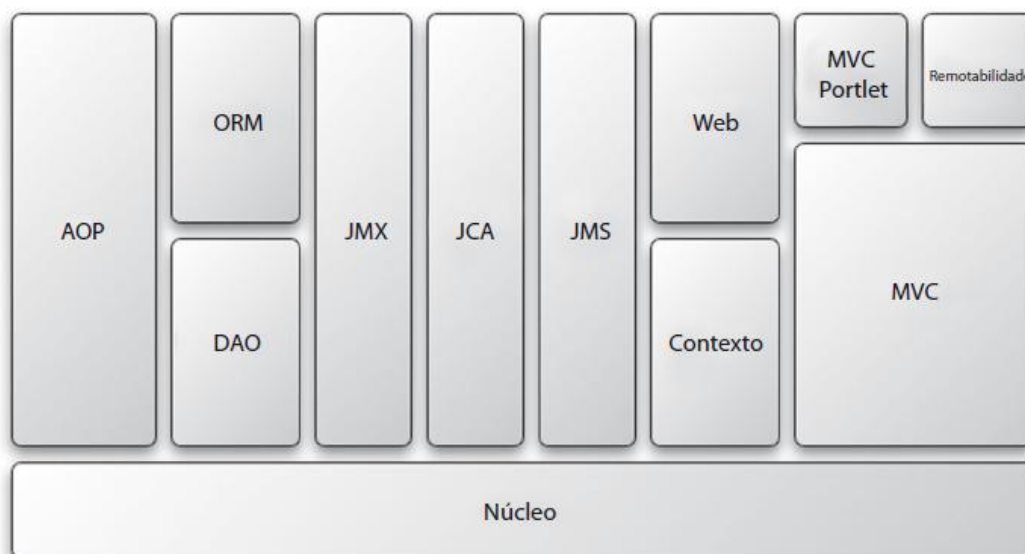


Figura 5. Módulos do Spring Framework.

Fonte : WALLS (2008)

Um problema particular no desenvolvimento de software são os códigos de copia e cola, uma característica do Spring e utilizar injeção de dependência, exemplo uma classe de conexão, utilizada em todos as classes que necessitarem dela, porém quando muda de ambiente a classe tem que ser alterada, usando a injeção de dependência criar a classe de forma parametrizada podendo ser utilizada em qualquer ambiente.

O Spring é um framework open source que visa estabelecer um modelo de componentização desenvolvimento J2EE mais fácil que o proposto através dos EJBs. A motivação para utilizar o Spring, seus objetivos, é desenvolver aplicativos de alta qualidade rapidamente. (MINTER,2008).

O Spring é um framework de aplicativo ajuda a estruturar aplicações inteiras em uma maneira consistente, produtiva, reunindo o melhor das estruturas de camada única para criar uma arquitetura coerente.

De acordo com (MINTER,2008) *“Desde a implementação generalizada de aplicações J2EE, em 1999/2000, J2EE não foi um sucesso absoluto na prática.”*

Porem trouxe uma padronização ao núcleo da camada intermediária, como gerenciamento de transações. As aplicações J2EE são complexas e tem excesso de esforço para desenvolver e ainda tem um desempenho decepcionante.

Apesar do Spring ser aplicável em uma ampla gama de ambientes não são apenas aplicações server-side J2EE - a motivação original para o Spring foi o ambiente J2EE, oferecendo serviços valiosos para aplicações J2EE. Deixando a complexidade do desenvolvimento das aplicações para o modelo de negócio.

As aplicações J2EE tendem a conter quantidade excessiva de código que na maioria das vezes não fazem nada como try/catch para adquirir e liberar recursos JDBC.De acordo com (MINTER,2008) *“EJB foi concebido como uma forma de reduzir a complexidade na aplicação da lógica de negócio em aplicações J2EE, mas não conseguiu esse objetivo em prática.”*

Um bom framework deve ser flexível durante a execução do código gerado, deve ser configurável ao invés de mudar as classes geradas, um framework bem concebido pode oferecer abstração coerente apenas como um atalho no desenvolvimento de aplicações complexas durante o ciclo de vida de um projeto.

Reconhecendo a importância dos frameworks de sucesso de projetos J2EE, muitos desenvolvedores e as empresas têm tentado escrever suas próprias estruturas, com variados graus de sucesso. Numa minoria de casos, o desenvolvimento atingiu o objetivo desejado e reduziu significativamente os custos e melhorou da produtividade.

No entanto na maioria dos casos, o custo de desenvolvimento e manutenção de um framework se tornou um problema, e falhas de projeto surgiram no framework. De acordo com (MINTER, 2008) *“é muito preferível trabalhar com um*

*único, amplamente utilizado e testado framework, ao invés de implementar um em casa.”*

Não importa quão grande organização, não será possível atingir um grau de experiência de correspondência que está disponível para um produto que é amplamente utilizado em muitas empresas. Se o framework é de código aberto, há uma vantagem na medida em que é possível contribuir para novas funcionalidades e melhorias que podem ser adotadas.

O Spring Framework surgiu da experiência de usar J2EE sem armação, ou com uma mistura de frameworks. Segundo (MINTER, 2008) *“Spring tem como objetivo simplificar o modelo de programação, ao invés de esconder a complexidade por trás de uma camada complexa de ferramentas.”*

O Spring permite usufruir dos benefícios-chave do J2EE, minimizando a complexidade encontrada pelo código do aplicativo.

Segundo (MINTER, 2008)

A essência do Spring está na prestação de serviços para POJOS ( Plain Old Java importante em um ambiente J2EE, mas o código do aplicativo distribuído como POJOS é naturalmente reutilizável em uma variedade de ambientes de execução.

Um framework só pode ser tão bom quanto o modelo de programação que ele proporciona. Se um Framework impõe muitas exigências sobre o código de usá-lo, ele cria um bloqueio de entrada e restringe os desenvolvedores de maneira que podem não ser aprimorados. O desenvolvedor da aplicação, ao invés de designer framework, muitas vezes tem uma compreensão melhor de como o código deve ser escrito.

No entanto, o framework deve fornecer orientação no que diz respeito às boas práticas: Deve fazer a coisa certa fácil de fazer. Obtendo o direito mistura de constrangimento e liberdade é o principal desafio do projeto framework, que tem tanto de arte como uma ciência.

Spring é o mais popular e mais ambicioso dos frameworks leves. É o único a abordar todos os níveis arquitetônico de uma aplicação J2EE típico, e a única a oferecer uma gama abrangente de serviços, bem como um recipiente e leve. Os módulos do Spring estão demonstrados na figura 6.



Spring utiliza AOP sob o capô para entregar importantes fora dos serviços de-caixa, tais como gerenciamento de transações declarativas. Spring AOP também pode ser usado para implementar o código personalizado que seriam dispersos entre as classes de aplicação. Disponibiliza o conceito de aspectos através do framework AOP Alliance e Aspectj para integrar os componentes de negócio com os serviços enterprise.

A abstração de dados de acesso ORM Spring incentiva uma abordagem coerente de arquitetura para acesso a dados e fornece uma abstração singular e poderosa para implementá-lo. Spring prevê uma hierarquia rica de exceções de acesso a dados, independente de qualquer produto persistência particular.

Implementa o suporte para integração com frameworks de mapeamento objeto relacional. Neste módulo encontra-se classes que dão suporte ao Hibernate, JPA, iBatis ou TopLink. Essas classes podem ser configuradas nos arquivos de configuração do Spring assim a aplicação não precisa ter linhas de código configurando framework de mapeamento objeto relacional ficando assim o gerenciamento feito pelo Spring framework.

O Spring fornece uma camada de abstração sobre JDBC que é significativamente mais simples e menos propensa a erros do que usar JDBC quando você precisa usar o acesso baseado em SQL para bancos de dados relacionais.

O Spring fornece uma abstração de controle de transação que fornece um modelo de programação consistente em uma ampla gama de ambientes e é a base para na gestão do Spring de transação declarativa e programática.

O Spring disponibiliza classes utilitárias para integração com serviços Java EE. JMS apoio: Spring fornece suporte para enviar e receber mensagens JMS de maneira muito mais simples do que o previsto por padrão J2EE. JMX apoio: Spring suporta JMX gestão de objetos de aplicativo que configura.

Suporte para uma estratégia de testes abrangentes para desenvolvedores de aplicativos: Spring, não só contribui para facilitar um bom design, permitindo testes de unidade eficaz, mas fornece uma solução abrangente para testes de integração fora de um servidor de aplicativos.

Framework web MVC do Spring fornece recursos para implementação de aplicações web baseado com integração frameworks MVC e com uma implementação própria de MVC. Seu uso de instâncias comuns de vários segmentos

"controladores" é semelhante à abordagem do Struts, mas framework Spring web é mais flexível e se integra perfeitamente com o container IoC Spring. Todas as funcionalidades do Spring outros também podem ser usados com outros frameworks web, como Struts ou JSF(Java Server Faces) Portlets(JSR-168).

O *Remoting* do Spring fornece suporte para remoting POJO baseado em uma série de protocolos, incluindo RMI, IIOP, e Hesse, Burlap, protocolos e serviços da web.

### 3.1 SPRING PORTIFÓLIO

O Spring Portfólio complementa o Spring framework como ilustra a figura 7.

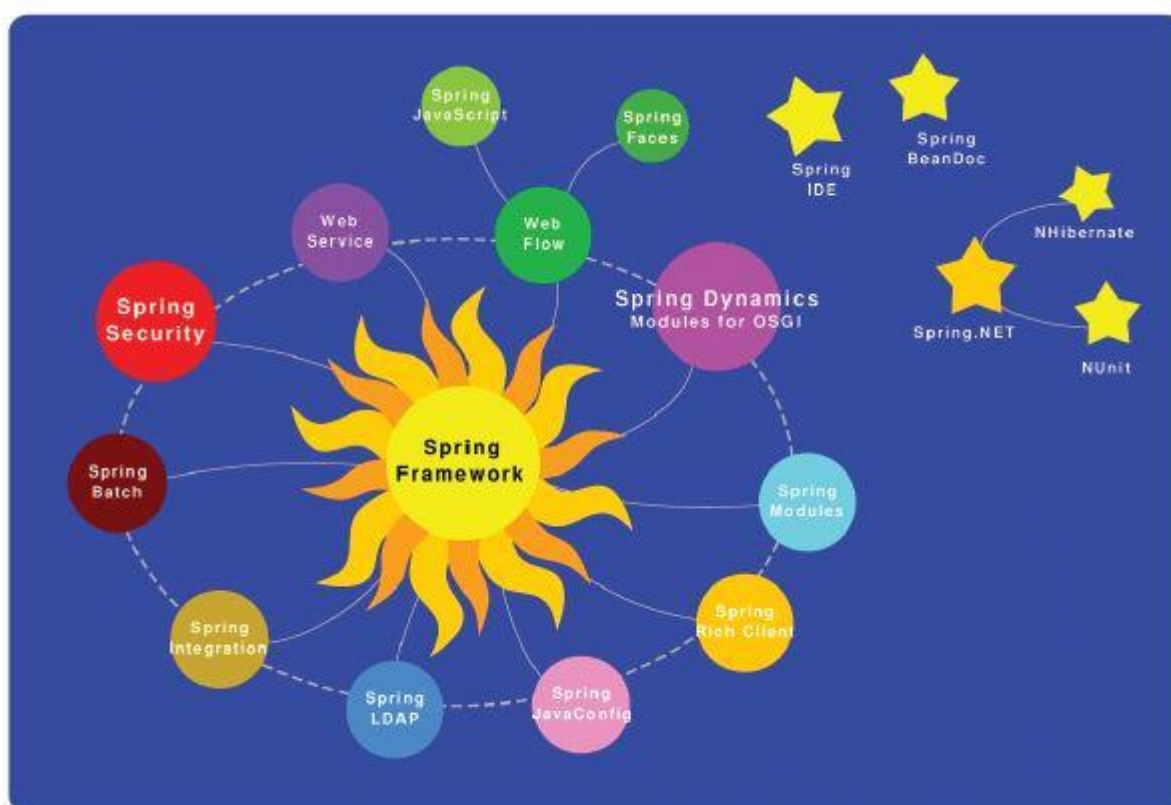


Figura 7. Spring Portfólio

Fonte: Lemos (2009)

O Spring Security trata autenticação e autorização, no nível de solução web e chamada de método, utilizando o Spring framework como base podendo usufruir de injeção de dependência e orientação a aspectos.

Os elementos vinculados ao Spring framework são pacotes distribuídos separadamente e disponíveis em um ou mais arquivos JAR. O Spring Security integrante do spring portfólio de acordo com (WALLS, 2008 )“*é um framework de segurança que fornece segurança declarativa a aplicações com base no Spring*”.

Tabela 1. Spring Portfólio

Pacotes	Descrição
<b>Spring Security</b>	Segurança declarativa via XML ou anotações com suporte a AOP e integração com tecnologias de segurança: JAAS, LDAP, DAO, OPENID, CAS, X509, Windows NTLM;
<b>Spring Web Services</b>	Suporte a web services a partir da definição do XML Schema e WSDL(Data Contract e Service Contract);
<b>Spring Web Flow</b>	Suporte ao controle de fluxos de navegação web, integração com JSF, conversação com AJAX;
<b>Dynamic Modules for OSGi</b>	Simplifica o uso da API OSGI através do Spring com POJO's;
<b>Spring Rich Cliente</b>	Recursos para desenvolvimento desktop com Swing e Spring;
<b>Spring Java Config</b>	Suporte à configuração dos beans programaticamente sem usar XML ou anotações;
<b>Spring LDAP</b>	Classes utilitárias dos beans para interação com um serviço de Lightweight Directory Access Protocol (LDAP).
<b>Spring Integration</b>	Implementa o suporte para interação de sistemas via mensagens (EAI e Enterprise Integration Patters);
<b>Spring Bath</b>	Suporte à execução de processos em batch de longa duração;
<b>Spring Bean Doc:</b>	Ferramenta para gerar documentação (semelhante ao Javadoc).
<b>Spring.NET</b>	Parte de parte do Spring Framework para desenvolvimento de aplicações .NET!

### 3.2 SPRING MVC

O Web framework Spring MVC é projetado para ajudar no desenvolvimento de aplicativos voltados para Web ajudando no gerenciamento de estados, fluxo de trabalho e validação. De acordo como (WALLS,2008 ).” O *Spring MVC é baseado no*

*padrão de projeto MVC (Model View Controller), que ajuda na construção de aplicações web flexíveis e de baixo acoplamento.”*

Segundo (WALLS, 2008) o funcionamento do Spring MVC é:

“Uma requisição processada toda vez que um usuário clica em um link ou envia um formulário do seu navegador da internet. A função de uma requisição parece com um mensageiro levando informação de um lugar para outro até o ponto que retorna uma resposta, deixando um pouco de informação e trazendo outras.”

A primeira parada é feita no DispatcherServlet do Spring, como na maioria dos frameworks MVC baseados em Java , o do Spring centraliza as requisições em um único servlet controller.(WALLS,2008)

A tarefa do DispatcherServlet é enviar a requisição a um controller do MVC Spring uma aplicação pode precisar de diversos controllers sendo assim o DispatcherServlet precisa ajudar a escolher para qual controller enviar a solicitação.(WALLS,2008)

O DispatcherServlet consulta um ou mais mapeamentos de controladores e determina a próxima parada da requisição e envia solicitação ao controller apropriado , no controller a solicitação deixa a sua carga e aguarda o seu processamento. A lógica executado pelo controller geralmente resulta em informações que precisam ser levadas de volta ao usuário e exibidas no navegador refere-se a essas informações como o model (modelo) .

A ultima tarefa do controller é agrupar os dados do modelo e o nome de uma view geralmente um JSP (Java Server Pages) precisando ser formatado de forma amigável tipicamente em HTML. A figura 8 extraída do livro (WALLS, 2008) demonstra o funcionamento do Spring MVC.

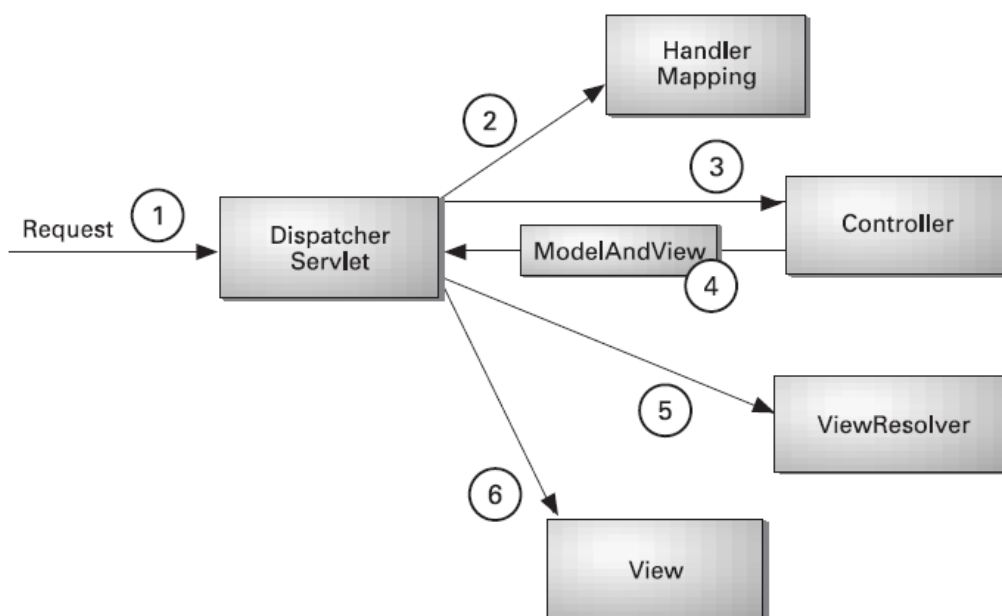


Figura 8. O ciclo de vida de uma requisição Spring MVC

Fonte : WALLS (2008)

Spring fornece suporte para diversos frameworks ORM (Mapeamento de objeto-relacional). Usar ferramentas ORM para camada de persistência pode reduzir milhares de linhas de códigos SQL e tempo de desenvolvimento.

O suporte ao Spring a frameworks ORM fornece, suporte integrado a transações declarativas do Spring, tratamento transparente a exceção, classes de modelos leves, classes com suporte a DAO e gerenciamento de recurso.

Nesse projeto foi utilizado a integração do o framework Hibernate por ser um framework de código aberto e de grande popularidade na comunidade de desenvolvedores Java. O Hibernate não fornece apenas mapeamento objeto relacional mas recursos sofisticados como cachê lazy load , eager fetching e cachê distribuído.

Hibernate é um framework ORM (Object Relational Mapping) mapeamento objeto relacional para persistência de dados de acordo com a documentação.

Foi iniciado em 2001 por Gavin King como uma alternativa ao uso de beans de entidade EJB2 estilo. Sua missão na época era

simplesmente oferecer melhores recursos de persistência que o oferecido por EJB2 simplificando as complexidades e permitindo recursos ausentes.

Hibernate é uma ferramenta que ajuda a persistir objetos Java em banco de dados relacional a partir da versão 3.2 do hibernate pode se substituir os metadados XML por anotações.

O trabalho do desenvolvedor é definir como os objetos serão mapeados nas tabelas do banco de dados, e o hibernate faz toda a conexão ao banco de dados , inclusive gera os comandos SQL.

## 4. ESTUDO DE CASO

Neste capítulo são apresentados a implementação e desenvolvimento de uma aplicação na qual demonstra os passos para o desenvolvimento utilizando o Spring framework, bem como a demonstração de algumas configurações.

A aplicação desenvolvida neste projeto realiza os casos de uso ilustrados na figura 9.

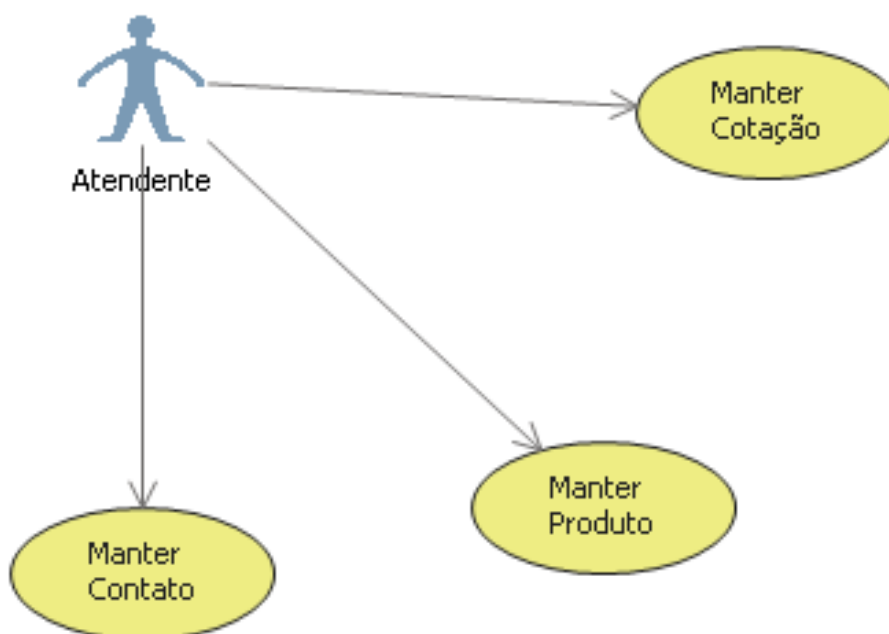


Figura 9. Modelo de caso de uso Manter Cotação.

Essa aplicação foi desenvolvida atendendo a requisitos: foi organizada seguindo o padrão de projeto MVC (Model View and Controller), os componentes de negócio são gerenciados pelo Spring Framework para suporte a transações e persistência, o uso do padrão de projeto DAO(Data Access Object) para encapsulamento das operações de persistência e interfaces Web foram implementadas utilizando Exts para permitir uma interface Rica para o usuários executarem os casos de uso.

## 4.1 PRÉ-REQUISITOS

A plataforma utilizada para o desenvolvimento da aplicação é o J2EE (Java 2 Enterprise Edition), utilizando o JDK (Java Development Kit) 1.6.0.16. Para servidor WEB utiliza-se apache e Tomcat 7.0.2 . A ferramenta IDE que auxilia o desenvolvimento da aplicações é o Eclipse Helios 2.6.2 rodando no sistema operacional Windows Vista ,juntamente com o browser Mozilla Firefox 3.5.11. A versão do framework Spring é 3.0.2 e do Banco de Dados MySql versão.

Como primeiro passo é criado um projeto Dinamic Web Project , através da opção File> New > Other – Dinamic Web Project na ferramenta Eclipse.

Após a criação do projeto a estrutura apresenta conforme demonstra a figura 10.

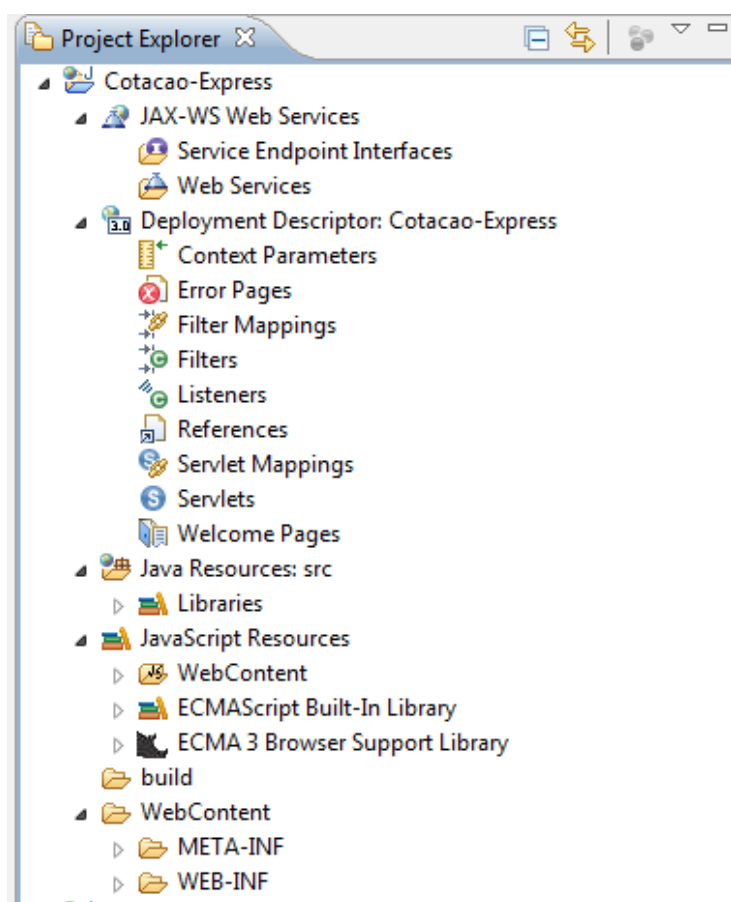


Figura 10. Estrutura do Dinamic Web Project

O segundo passo seguido para utilizar o Spring Framework nesse projeto foi fazer o seu download . O Spring framework pode ser obtido através da seguinte página: [HTTP://www.springframework.org/download/](http://www.springframework.org/download/).

A distribuição contém arquivos como spring.jar, commons-logging.jar que são obrigatórios para o funcionamento do Spring.

O projeto CotacaoExpress recebe os jars conforme mostra a figura 11.

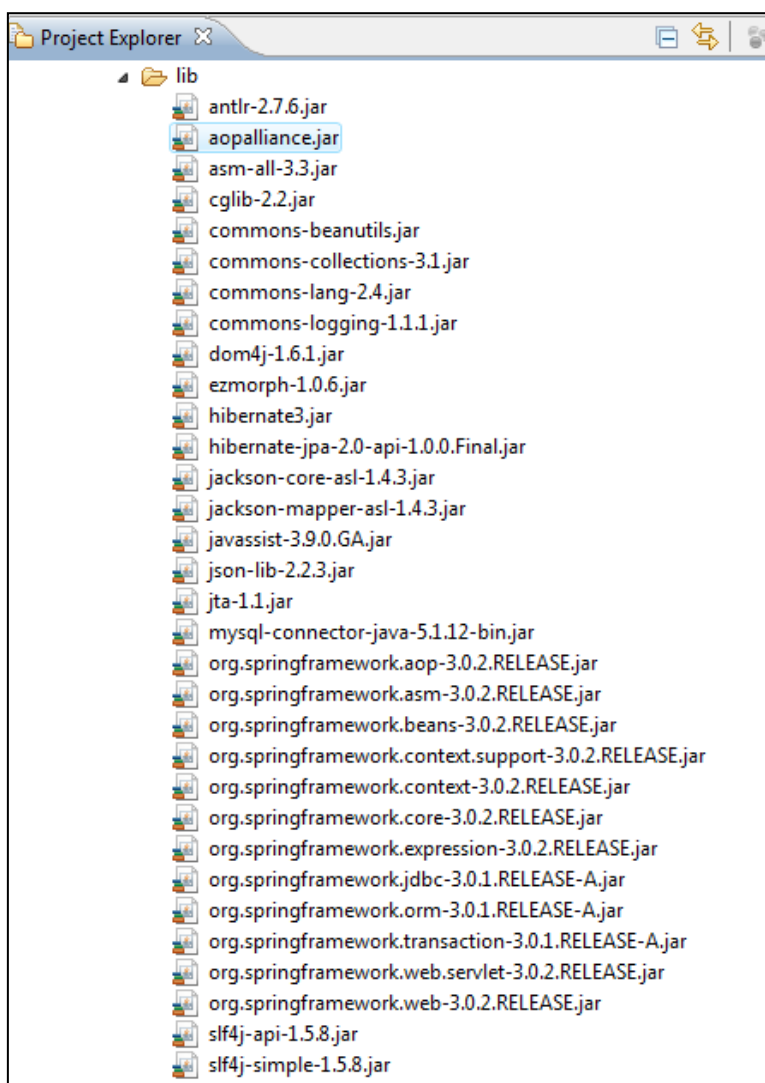


Figura 11. Pasta lib

No projeto foi criada uma pasta WEB-INF na pasta do mesmo uma pasta chamada lib, no qual é destinada a receber as bibliotecas dos frameworks utilizados os arquivos jars mostrados na figura 11 . Todos os jars da pasta devem ser adicionado são classpath do projeto.

## 4.2 MODELOS DE NEGÓCIO

Os modelos de negócio utilizadas por esse componente Cotação Express foram implementados como Java-Beans. Objetos destas classes tem seus atributos armazenados em um banco de dados relacional para persistência das informações. A figura 12 apresenta o diagrama de classes do projeto.

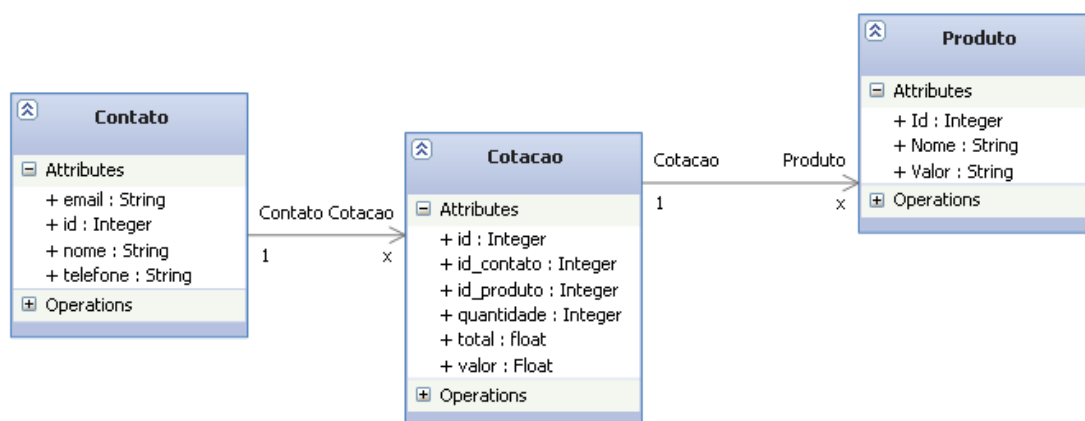


Figura 12. Diagrama de Classes do Projeto.

A modelagem de dados para este projeto é bastante simples, apenas três classes, onde um contato pode ter várias cotações e uma cotação pode ter vários produtos, e uma cotação só pode ter um contato.

Utilizando JPA e o framework hibernate como tecnologia de persistência de dados, é necessário utilizar algumas anotações na classe que descreve as entidades do sistema como `@Entity`. Para indicar a chave primária, existe a anotação `@Id` e para gerar chave primária automática usa-se a anotação `@GeneratedValue`, como demonstrado na figura 13.

```
package com.carol.model;
import javax.persistence.Column;
@JsonAutoDetect
@Entity
@Table(name = "PRODUTO")
public class Produto {

    private int id;
    private String nome;
    private long valor;

    @Id
    @GeneratedValue
    @Column(name="PRODUTO_ID")
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @Column(name="PRODUTO_NOME", nullable=false)
    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    @Column(name="PRODUTO_VALOR", nullable=false)
    public long getValor() {
        return valor;
    }

    public void setValor(long valor) {
        this.valor = valor;
    }
}
```

Figura 13. Classe Produto

A classe Contato representa um contato com um id ,um nome, um email e um phone. Já a classe Produto representa um produto com id, nome e valor. Os objetos dessa classe são gerenciados pelo Spring para as entidades de negócio.

### 4.3 CAMADA DE PERSISTÊNCIA DE DADOS

Nas operações de persistência serão encapsuladas em um DAO para cada entidade estes DAOs serão gerenciados pelo Spring.

DAO(Data Access Object) é um padrão de projeto de software que permite que outras partes do sistema utilize as funcionalidades de acesso a dados de forma desacoplada. A figura 14 demonstra a classe ContatoDAO para definição dos serviços DAOs e objetos responsáveis pela persistência da entidade Contato.

Esta operações do tipo persistência podem produzir exceções do Tipo DAOServiceException ou retornar o valor desejado como indicado na assinatura dos métodos.

A interface IContatoDAO conforme demosntrado na figura 15 implementa o DAO para persistência de objetos do Tipo Contato com a chave primária representada por um Integer. Nessa interface foram adicionados métodos de persistência específicos para contato, como por exemplo, a operação de busca de contato pelo nome.

```

package com.carol.dao;
import java.util.List;

@Repository
public class ContactDAO implements IContactDAO{

    private HibernateTemplate hibernateTemplate;

    @Autowired
    public void setSessionFactory(SessionFactory sessionFactory) {
        hibernateTemplate = new HibernateTemplate(sessionFactory);
    }
    @SuppressWarnings("unchecked")
    @Override
    public List<Contact> getContacts() {
        return hibernateTemplate.find("from Contact");
    }
    @Override
    public void deleteContact(int id){
        Object record = hibernateTemplate.load(Contact.class, id);
        hibernateTemplate.delete(record);
    }
    @Override
    public Contact saveContact(Contact contact){
        hibernateTemplate.saveOrUpdate(contact);
        return contact;
    }
}

```

Figura 14. ContactDAO

```

package com.carol.dao;

import java.util.List;
import com.carol.model.Contact;

public interface IContactDAO {

    List<Contact> getContacts();

    void deleteContact(int id);

    Contact saveContact(Contact contact);

}

```

Figura 15. Interface IContactDAO.

A implementação realizada pelas classes Contato, Produto e Cotação foram simples utilizando o contexto de persistência e os DAOs instanciados pelo Spring através de métodos setters ou anotações.

A anotação @Transaction sobre os métodos, indica para o Spring que esse método deve ser executado sob uma transação. Se algum outro componente invocar esse método anotado, o Spring criará uma transação e efetivará esta transação ao final da execução com o sucesso desse método.

A implementação deste DAO tem um atributo chamando entityManager baseado no gerenciador de entidades JPA (Java Persistence API). O Spring Framework irá injetar uma referencia destes objetos no atributo do DAO.

Apesar o Spring ter capacidade de detectar anotações ainda foi necessário definir um XML de configuração.

A figura 16 mostra o arquivo de configuração do Spring, neste arquivo está configurado um objeto de conexão com o banco de dados MYSQL. O dataSource (blog) a ser criado pelo Spring contém configurações que definem a classe de driver JDBC, a ORM de conexão , o usuário e a senha. O Spring criará um gerenciador de transações (txManager) e uma fábrica de EntityManagers baseada no JPA.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:tx="http://www.springframework.org/schema/tx"
      xmlns:aop="http://www.springframework.org/schema/aop"
      xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
        http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">

  <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName"><value>com.mysql.jdbc.Driver</value></property>
    <property name="url"><value>jdbc:mysql://localhost/blog</value></property>
    <property name="username"><value>edson</value></property>
    <property name="password"><value>integrator</value></property>
  </bean>

  <!-- Hibernate SessionFactory -->
  <bean id="sessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
    <property name="dataSource"><ref local="dataSource"/></property>
    <property name="packagesToScan" value="com.carol.model" />
    <property name="hibernateProperties">
      <props>
        <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
        <prop key="hibernate.show_sql">false</prop>
        <prop key="hibernate.hbm2ddl.auto">update</prop>
      </props>
    </property>
  </bean>

  <!-- Transaction manager for a single Hibernate SessionFactory (alternative to JTA) -->
  <tx:annotation-driven/>
  <bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory"><ref local="sessionFactory"/></property>
  </bean>
```

Figura 16. DBconfig.xml

A classe ProdutoService foi criada para integrar com a EXTJs criando um objeto JSON conforme demonstra a figura 17.

A anotação @Service é usada para identificar o componente de negócio gerenciado pelo Spring.

Para que a injeção de dependência seja utilizada automaticamente a anotação @Autowired foi inserida.

```

package com.carol.service;
import java.util.ArrayList;
@Service
public class ProdutoService {
    private ProdutoDAO produtoDAO;
    private Util util;
    @Transactional(readOnly=true)
    public List<Produto> getProdutoList(){
        return produtoDAO.getProdutos();
    }
    @Transactional
    public List<Produto> create(Object data){
        List<Produto> newProdutos = new ArrayList<Produto>();
        List<Produto> list = util.getProdutosFromRequest(data);
        for (Produto produto : list){
            newProdutos.add(produtoDAO.saveProduto(produto));
        }
        return newProdutos;
    }
    @Transactional
    public List<Produto> update(Object data){
        List<Produto> returnProdutos = new ArrayList<Produto>();
        List<Produto> updatedProdutos = util.getProdutosFromRequest(data);
        for (Produto produto : updatedProdutos){
            returnProdutos.add(produtoDAO.saveProduto(produto));
        }
        return returnProdutos;
    }
    @Transactional
    public void delete(Object data){
        if (data.toString().indexOf(',') > -1){
            List<Integer> deleteProdutos = util.getListIdFromJSON(data);
            for (Integer id : deleteProdutos){
                produtoDAO.deleteProduto(id);
            }
        }
        else {
            Integer id = Integer.parseInt(data.toString());

            produtoDAO.deleteProduto(id);
        }
    }
    @Autowired
    public void setProdutoDAO(ProdutoDAO produtoDAO) {
        this.produtoDAO = produtoDAO;
    }
    @Autowired
    public void setUtil(Util util) {
        this.util = util;
    }
}

```

Figura 17 . Classe Produto Service

## 4.4 CAMADA LÓGICA E CONTROLE

Os controladores gerenciam qual tela deve aparecer quando se clica em uma opção na tela inicial conforme demonstra a figura 17, foram criados três controladores:

ContatoController – Mostra o formulário de cadastro de contatos, apaga, salva e altera contatos.

ProdutoController - Mostra o formulário de cadastro de produtos, apaga, salva e altera produtos.

A anotação `@Controller` é usada para dizer ao Spring que a classe é um Controller, e `@RequestMapping` indica que vai atender as requisições `/produto` na URL combinando o Value do `@RequestMapping` com o nome da Classe existente.

```
import java.util.HashMap;

@Controller
public class ProdutoController {

    private ProdutoService produtoService;

    @RequestMapping(value="/produto/view.action")
    public @ResponseBody Map<String,? extends Object> view() throws Exception {

        try{

            List<Produto> produtos = produtoService.getProdutoList();

            return getMap(produtos);

        } catch (Exception e) {

            return getModelMapError("Error retrieving Produtos from database.");

        }

    }

    @RequestMapping(value="/produto/create.action")
    public @ResponseBody Map<String,? extends Object> create(@RequestParam Object data) throws Exception {

        try{

            List<Produto> produtos = produtoService.create(data);

            return getMap(produtos);

        } catch (Exception e) {

            return getModelMapError("Error trying to create produto.");

        }

    }

}
```

Figura 18. Produto Controller

A instrução para o Spring MVC Serializaro hashmap para o cliente é feita através da anotação @ResponseBody.

CotacaoController – Mostra o formulário de cadastro da cotação , seleciona o contato , seleciona os produtos e soma o valor da cotação.

## 4.5 CAMADA DE APRESENTAÇÃO

A camada de apresentação possui uma tela inicial para acessar a aplicação. A figura 19 demonstra o layout da tela inicial do cotação express .



Figura 19. Tela inicial do Cotação Express.

Para a aplicação foram criadas três telas Produto , Contato e Cotação. As telas foram criadas em arquivos JSP, que executam um script ExtJs para montar a tela na figura 20 é demonstrado parte do código javascript para geração da tela.

O Spring MVC automaticamente serializa o objeto para JSON porque o cliente (Browser) aceita esse tipo de conteúdo.

Utilizando a função `Ext.data.HttpProxy` passando o mapeamento da classe mais o método a interface através de uma API de desenvolvimento realiza as operações de leitura (read) , criação (create) , alteração (update) e exclusão (destroy).

```
Ext.BLANK_IMAGE_URL = '/extjs-crud-grid/ext-3.2.1/resources/images/default/s.gif';

var Produto = Ext.data.Record.create([
{name: 'id'},
{
    name: 'name',
    type: 'string'
}, {
    name: 'valor',
    type: 'long'
}]);

var proxy = new Ext.data.HttpProxy({
    api: {
        read : 'produto/view.action',
        create : 'produto/create.action',
        update: 'produto/update.action',
        destroy: 'produto/delete.action'
    }
});

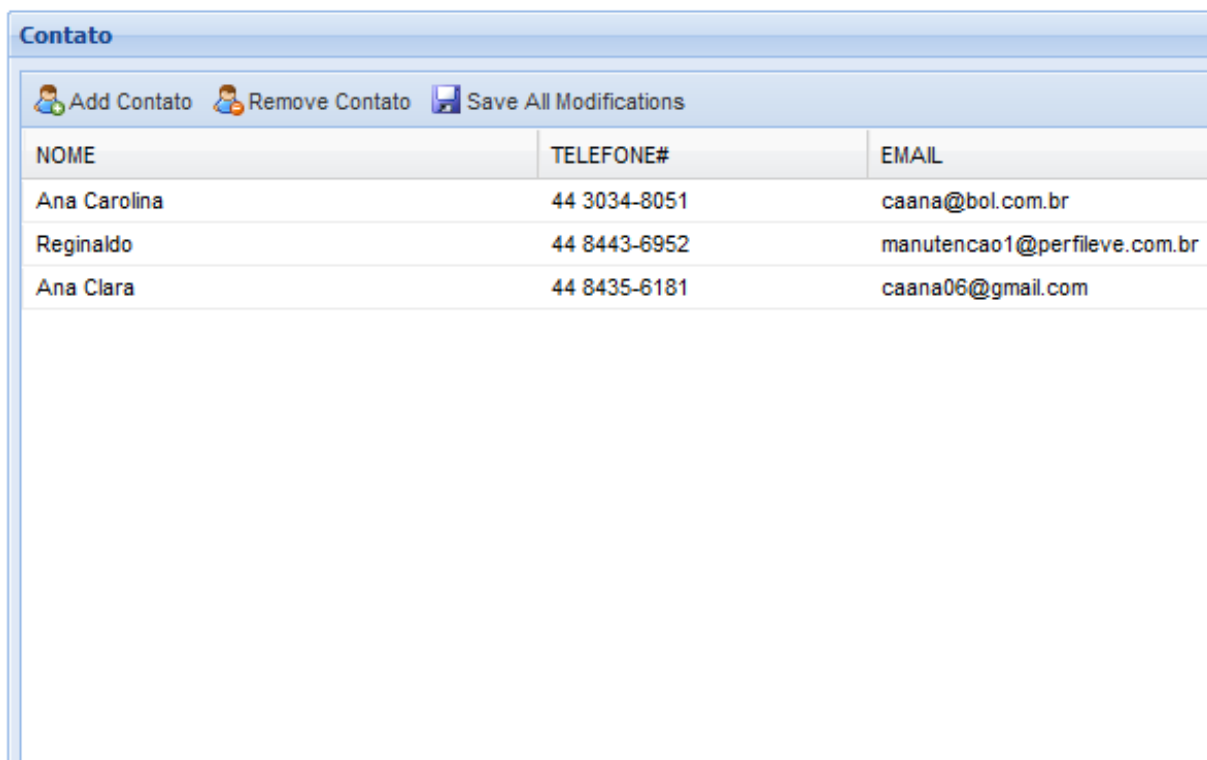
var reader = new Ext.data.JsonReader({
    totalProperty: 'total',
    successProperty: 'success',
    idProperty: 'id',
    root: 'data',
    messageProperty: 'message' // <-- New "messageProperty" meta-data
},
Produto);
```

Figura 20. Parte do código Java Script para montagem da tela cadastro de Produto.

## 5. RESULTADOS

O estudo de caso foi desenvolvido e está em fase de avaliação de viabilidade para implantação de um sistema Web utilizando o framework spring junto com outros frameworks como o Hibernate e ExtJs.

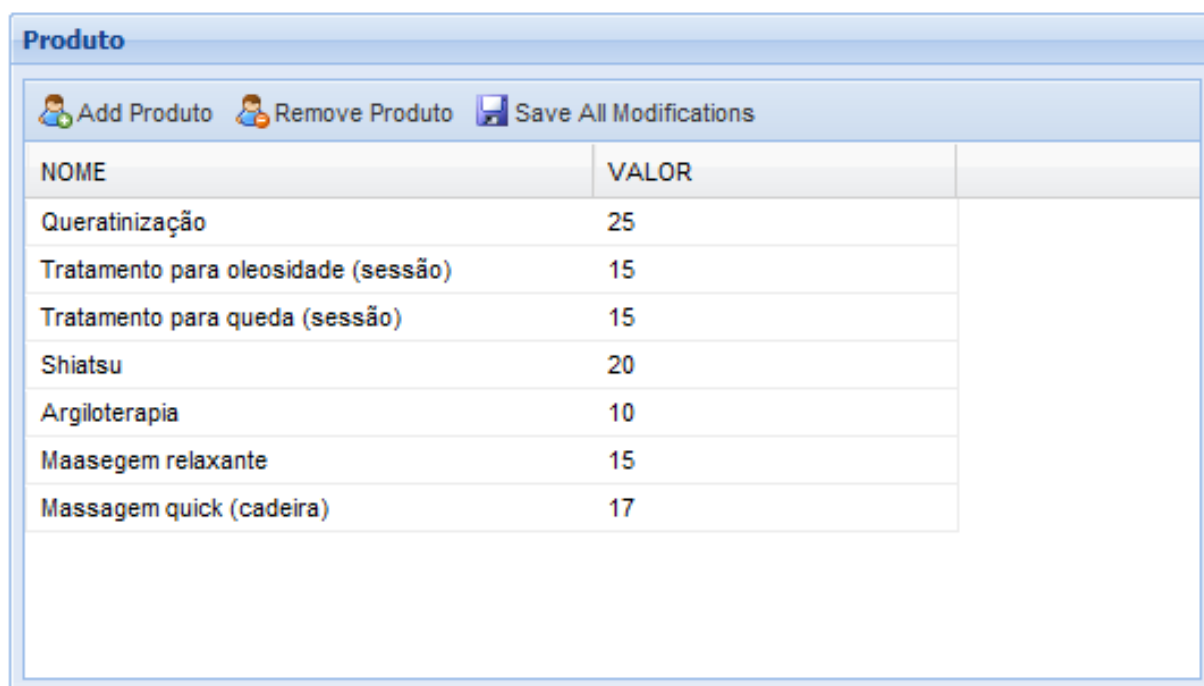
A tela de cadastro de Clientes pode se cadastrar os dados para contato com o cliente como nome , telefone e email.Nesta etapa somente são validados se os dados estão sendo preechidos.O resultado da tela de Cliente está demonstrado na figura 19.



NOME	TELEFONE#	EMAIL
Ana Carolina	44 3034-8051	caana@bol.com.br
Reginaldo	44 8443-6952	manutencao1@perfileve.com.br
Ana Clara	44 8435-6181	caana06@gmail.com

Figura 21. Tela de Cadastro de Contato

Para que os clientes possam ser atendidos a clinica de estética necessita ter produtos/serviços a oferecer aos clientes, tendo que se cadastrar os produtos com descrição e valor de cada produto. A tela de cadastro de produto é demonstrada na figura 20.



The screenshot shows a web interface titled "Produto". At the top, there are three buttons: "Add Produto" (with a person icon), "Remove Produto" (with a person and minus icon), and "Save All Modifications" (with a floppy disk icon). Below these buttons is a table with two columns: "NOME" and "VALOR". The table contains the following data:

NOME	VALOR
Queratinização	25
Tratamento para oleosidade (sessão)	15
Tratamento para queda (sessão)	15
Shiatsu	20
Argiloterapia	10
Maasegem relaxante	15
Massagem quick (cadeira)	17

Figura 22. Tela de Cadastro de Produto

O Cotação Express surgiu da idéia de agilizar o atendimento aos clientes fazendo uma cotação rápida. O resultado da tela de cotação está demonstrado na figura 13.

Produtos Selecionados			
Produto	Quantidade	Valor Unitário	Valor Total
Tratamento para oleosidade (sessão)	4	R\$15.00	R\$60.00
Maasegem relaxante	6	R\$15.00	R\$90.00
	10	R\$15.00	R\$150.00

Figura 23. Tela de cadastro da Cotação.

O desenvolvimento utilizando ferramentas gratuitas e frameworks open-source embora sem investimento inicial requer um grande esforço das equipes de desenvolvimento para o aprendizado da nova tecnologia, e por ser muito recente a dificuldade em encontrar treinamentos e material é grande.

No cotação Express são realizadas algumas validações, não pode-se cadastrar contato sem nome, email e telefone. No cadastro de Produto é obrigatório o preenchimento do nome e valor do produto. Na cotação é necessário solicitar um cliente ao menos um produto como demonstra a figura 24 e também que informar a quantidade a ser solicitada.

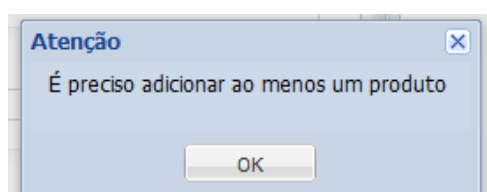


Figura 24. Validação Produto na cotação.

Resultado final o Spring Framework facilita o desenvolvimento de aplicações podendo ser utilizado tanto para aplicações de pequeno porte quanto aplicações de grande porte até aplicações distribuídas sendo um framework para desenvolvimento de uma aplicação completa.

## 6. CONCLUSÃO

Foram utilizadas várias tecnologias disponíveis integradas ao framework Spring , por estar utilizando EXTJs nas telas de cadastro utilizei o tomcat 7.02 por não funcionar com outra versão do tomcat.

Utilizando os recursos de autocompletar, compilação a ferramenta Eclipse como ambiente de desenvolvimento e o tomcat 7.02 facilitaram o desenvolvimento e os testes.

De uma maneira geral o desenvolvimento utilizando o framework Spring no inicio pode ser trabalhoso, porém com conhecimento das integrações das facilidades que o framework disponibiliza é possível afirmar que o otimiza o desenvolvimento, qualidade da codificações , na facilidade para manutenção e principalmente na estrutura que o Spring implementa deixando somente se preocupar com a aplicação.

A elaboração do Cotação Express pode ser considerada parcialmente cumprida, pois funcionalidades consideradas importantes pelos atendentes como agendamento de atendimento , atendentes para que o sistema fosse posta em pratica não foram implementadas a tempo de concluir esse trabalho. Tal fato se deve principalmente ao escopo deste trabalho não está atender a todos os requisitos que o sistema necessita para substituir a forma atual de atendimento.

Todo o desenvolvimento do sistema foi realizado utilizando os princípios do Spring, a qual possibilitou um desenvolvimento ágil. A utilização de práticas como o desenvolvimento um projeto simples se mostraram muito benéficas para a produtividade e a qualidade do código desenvolvido. O sistema desenvolvido possui uma arquitetura simples, na qual foram necessárias poucas alterações durante o desenvolvimento, sendo a maior parte das alterações referentes a telas do sistema.

Finalmente, o sistema está parcialmente concluído estão prontos os cadastros de clientes e produtos e a visualização da cotação para ser utilizado em clinicas de estética para atendimento aos clientes.

Para trabalhos futuros pretende-se dar continuidade ao desenvolvimento do sistema, implementando novas funcionalidades, como controle de agendamento e possibilidade de impressão do comprovante de agendamento, ficha de controle do cliente e um elaborado sistema de fidelidade de clientes.

## 7. REFERÊNCIAS

BAUER, Christian; KING, Gavin. Hibernate in action. Greenwich: Manning Publications,2005.

DEITEL, Harvey M. , DEITEL Paul J. Java Como Programar, 6.ed. Person Education 2005 p.5.

GARCIA, Jesus , 2010. ExtJS In Action , GRENWICH , CT USA Manning 2010, p.3 -4.

GOETTEN Junior, WINCK Diogo, 2006. AspectJ – Programação Orientada a Aspectos com Java. São Paulo – SP: Novatec Editora, 2006 p.41 -51.

Hibernate Reference Documentation [internet] 2010 acesso out 24. Disponível em : [www.hibernate.org](http://www.hibernate.org)

JSON Reference Documentation [internet] 2010 acesso out 23. Disponível em :[www.json.org](http://www.json.org)

LEMOS , revista java magazine edição 65 – Artigo Criando uma aplicação Web com Spring – Alberto J.

MINTER, Dave, 2008. Beginning to Spring – From Novice to Professional. New York-NY. APRESS 2008 p.1.

Spring Framework Reference Documentation[internet] 2010 acesso em out 22. Disponível em: [www.springframework.org](http://www.springframework.org)

STONEBRAKER, M. *Object-Relational Database Systems*. Morgan Kaufman,1996

VUKOTIC , Aleksa MACHACEK Jan, 2009. Pro Spring 2.5. São Paulo-SP: Editora Ciência Moderna 2009 p.3 -5 e 38.

WALLS, Craig , BREIDENBACH, Ryan. Spring in Action , GRENWICH , CT USA Manning 2008 p.5-6 e 278.