

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS PATO BRANCO
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE
SISTEMAS**

CARLOS ALEXANDRE RASADOR DA SILVA

APLICAÇÃO RIA PARA GERENCIAMENTO DE OFICINA MECÂNICA

TRABALHO DE CONCLUSÃO DE CURSO

**PATO BRANCO
2013**

CARLOS ALEXANDRE RASADOR DA SILVA

APLICAÇÃO RIA PARA GERENCIAMENTO DE OFICINA MECÂNICA

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

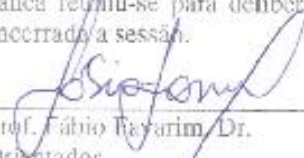
Orientador: Prof. Dr. Fábio Favarim

**PATO BRANCO
2013**


ATA Nº: 205

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO CARLOS ALEXANDRE RASADOR DA SILVA.

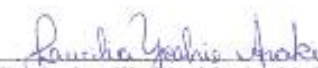
Às 10:30 hrs do dia 17 de abril de 2013, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Fábio Favarim (Orientador), Beatriz Terezinha Borsari (Convidada) e Lucilia Yoshie Araki (Convidada), para avaliar o Trabalho de Diplomação do aluno Carlos Alexandre Rasador da Silva, matrícula 603589, sob o título **Aplicação RIA para Gerenciamento de Oficina Mecânica**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 11:20 hrs foi encerrada a sessão.




Prof. Fábio Favarim, Dr.
Orientador




Profa. Beatriz Terezinha Borsari, Dr.
Convidada



Profa. Lucilia Yoshie Araki, M.Sc.
Convidada



Prof. Omeiro Francisco Bernol, M.Sc.
Coordenador do Trabalho de Diplomação



Prof. Edilson Puntarolo, Dr.
Coordenador do Curso

RESUMO

SILVA, Carlos Alexandre Rasador. Aplicação RIA para gerenciamento de oficina mecânica. 2013. 56f. Monografia de Trabalho de Conclusão de Curso. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2013.

Neste trabalho de conclusão de curso são apresentadas a análise, o projeto e o desenvolvimento de um sistema para gerenciamento de oficina mecânica. Com o intuito de atender a um cliente do acadêmico, e também como forma de aprendizado de novas tecnologias, foi desenvolvido uma Aplicação Rica para Internet (RIA) utilizando como base o padrão de desenvolvimento MVC (*Model View Controller*). O sistema permite o controle dos serviços prestados, controle das peças vendidas, controle do estoque de peças e controle do contato de clientes e fornecedores, entre outros.

Palavras-chave: RIA. MVC. Oficina Mecânica. Sistema Web.

ABSTRACT

SILVA, Carlos Alexandre Rasador. RIA application for managing mechanic shop. 2013. 56f. Monografia de Trabalho de Conclusão de Curso. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2013.

In this work of completion are presented the analysis, the design and development of a system for managing mechanic shop. In order to cater to a customer's academic, and also as a way of learning new technologies, has developed a Rich Internet Application (RIA) utilizing as the standard base development MVC (*Model View Controller*). The system allows control of services, control of parts sold, control of parts inventory and control of customers and suppliers, among others.

Keywords: RIA. MVC. Mechanic Shop. Web System.

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicidade, Consistência, Isolamento, Durabilidade
AJAX	<i>Asynchronous Javascript And XML</i>
AMF	<i>Action Message Format</i>
BIRT	<i>Business Intelligence and Reporting Tools</i>
CSS	<i>Cascading Style Sheets</i>
DNS	<i>Domain Name System</i>
EJB	<i>Enterprise JavaBeans</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IBM	<i>International Business Machines</i>
IDE	<i>Integrated Development Environment</i>
IIS	<i>Internet Information Services</i>
IP	<i>Internet Protocol</i>
J2EE	<i>Java 2 Enterprise Edition</i>
JDBC	<i>Java Database Connectivity</i>
JNDI	<i>Java Naming and Directory Interface</i>
JSP	<i>Java Server Pages</i>
JVM	<i>Java Virtual Machine</i>
MXML	<i>Minimal XML</i>
MVC	<i>Model View Controller</i>
PDF	<i>Portable Document Format</i>
PHP	<i>PHP Hypertext Preprocessor</i>
RIA	<i>Rich Internet Application</i>
SQL	<i>Structured Query Language</i>
UML	<i>Unified Modeling Language</i>
UTFPR	Universidade Tecnológica Federal do Paraná
XML	<i>Extensible Markup Language</i>

LISTA DE FIGURAS

Figura 1 – Interações entre cliente e servidor	12
Figura 2 – RIA combina o melhor do <i>Desktop</i> , da Web e de Comunicações	14
Figura 3 – Interações entre cliente, camada adicional e servidor.....	17
Figura 4 – Visões ligadas a um modelo	19
Figura 5 - Eclipse IDE sob a perspectiva Java	24
Figura 6 - Flash Builder 4 na perspectiva Flash	25
Figura 7 – Comparativo do protocolo AMF e XML	27
Figura 8 - Componente de criação de relatório do BIRT	28
Figura 9 – Diagrama de casos de uso.....	32
Figura 10 - Diagrama de classes.....	33
Figura 11 - Diagrama de entidade relacionamento	34
Figura 12 - Tela de autenticação do sistema.....	35
Figura 13 - Caixa de mensagem de validação	35
Figura 14 - Menu de opções do sistema	35
Figura 15 - Tela de clientes	36
Figura 16 - Formulário de cadastro de cliente	37
Figura 17 - Mensagem de confirmação de sucesso	37
Figura 18 - Filtro de pesquisa de clientes.....	38
Figura 19 - Tela de fornecedores	38
Figura 20 - Tela de produtos	39
Figura 21 - Formulário de cadastro de produto	40
Figura 22 - Filtro de pesquisa de produtos.....	40
Figura 23 - Tela de estoque	41
Figura 24 - Tela de veículos	42
Figura 25 - Formulário de cadastro de veículo	42
Figura 26 - Filtro de pesquisa de veículos.....	43
Figura 27 - Tela de serviços.....	44
Figura 28 – Formulário de cadastro de serviço	45
Figura 29 - Filtro de pesquisa de serviços.....	46
Figura 30 - Pacotes e classes do projeto em Java	47
Figura 31 - Arquivo de configuração de persistência de dados.....	47
Figura 32 - Classe conectar	48
Figura 33 - Utilização dos métodos da classe conectar	48
Figura 34 - Arquivo remoting-config.xml.....	49
Figura 35 - Classe conexao.....	50
Figura 36 - Endereço IP do sistema	51

LISTA DE QUADROS

Quadro 1 - Comparação entre <i>Desktop</i> incluindo cliente e servidor, aplicações Web e RIA	15
Quadro 2 - Lista de ferramentas e tecnologias utilizadas.....	21
Quadro 3 – Requisitos funcionais.....	31
Quadro 4 – Requisitos não funcionais.....	31
Quadro 5 - Descrição dos casos de uso	32

SUMÁRIO

1	INTRODUÇÃO	7
1.1	CONSIDERAÇÕES INICIAIS	7
1.2	OBJETIVOS	8
1.2.1	Objetivo geral	8
1.2.2	Objetivos específicos	9
1.3	JUSTIFICATIVA.....	9
1.4	ORGANIZAÇÃO DO TRABALHO.....	10
2	REFERENCIAL TEÓRICO	11
2.1	APLICAÇÕES WEB TRADICIONAIS	11
2.2	APLICAÇÕES RICAS PARA INTERNET (RIA)	13
2.2.1	Motivação para Aplicações RIA	13
2.2.2	Características de RIA	15
2.2.3	Funcionamento de RIA.....	16
2.2.4	Tecnologias RIA existentes.....	17
2.3	ARQUITETURA DE SOFTWARE MVC	18
3	MATERIAIS E MÉTODO	20
3.1	MATERIAIS	20
3.1.1	Linguagem de Programação Java.....	21
3.1.2	Apache Flex	22
3.1.3	Eclipse IDE.....	23
3.1.4	Flash Builder	24
3.1.5	Apache Tomcat	25
3.1.6	Adobe BlazeDS	26
3.1.7	BIRT	27
3.1.8	MySQL	28
3.2	MÉTODO	29
4	RESULTADOS.....	30
4.1	APRESENTAÇÃO DO SISTEMA	30
4.2	REQUISITOS.....	30
4.3	DIAGRAMA DE CASOS DE USO	31
4.4	DIAGRAMA DE CLASSES	33
4.5	DIAGRAMA DE ENTIDADE RELACIONAMENTO	33
4.6	DESCRIÇÃO DO SISTEMA	34
4.6.1	Autenticação no Sistema.....	34
4.6.2	Cadastro de Clientes.....	36
4.6.3	Cadastro de Fornecedores.....	38
4.6.4	Cadastro de Produtos	39
4.6.5	Cadastro de Veículos	41
4.6.6	Serviços	43
4.7	IMPLEMENTAÇÃO DO SISTEMA.....	46
4.8	TESTES DO SISTEMA.....	50
4.9	IMPLANTAÇÃO DO SISTEMA	51
5	CONCLUSÃO.....	53
	REFERÊNCIAS BIBLIOGRÁFICAS	54

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais do trabalho, com o contexto da aplicação do sistema desenvolvido, os seus objetivos a justificativa e a organização do texto.

1.1 CONSIDERAÇÕES INICIAIS

Cada vez mais as organizações, sejam estas empresas, instituições de ensino, entidades de classe, buscam substituir os processos executados diariamente de forma manual por sistemas automatizados. A utilização destes sistemas visa reduzir custos e agilizar consideravelmente esses processos.

Com o crescente uso de sistemas computacionais, assim como o desenvolvimento das redes de computadores, surgiu uma nova tecnologia: a tecnologia Web. A Web surgiu como um repositório de divulgação de informação. Ao longo do tempo essa tecnologia foi sendo modificada de forma a incorporar novos recursos e funções. Entre estas funções está o desenvolvimento de aplicações Web. As aplicações Web consistem de uma arquitetura Cliente-Servidor, na qual todo o processamento e o armazenamento dos dados ficam no servidor, enquanto o cliente consiste em uma página Web estática. Neste tipo de sistema, a interação com a aplicação é feita através do *browser*, no qual as informações são enviadas para o servidor, estes as processa e responde ao cliente. Quando o cliente recebe os dados processados a página é recarregada.

Com o uso crescente da *Web*, seja pelo acesso a um grande repositório de informações assim como de aplicações *Web*, proporcionou uma nova geração desta tecnologia, a chamada *Web 2.0* (DEITEL, 2008). A *Web 2.0* faz uso de novas ferramentas e tecnologias a fim de proporcionar uma série de novas características às aplicações. Uma das tecnologias mais difundidas atualmente que fazem parte dessa nova geração da *Web* são as RIAs (*Rich Internet Application*) ou Aplicações Ricas para Internet (BOZZON, 2006). As aplicações RIA transferem todo o processamento da interface para o *browser*, no entanto, o processamento dos

dados, assim como o armazenamento destes ficam no servidor. As aplicações RIAs possuem características e funcionalidades de aplicações tradicionais do tipo *desktop*, proporcionando maior ergonomia, usabilidade, interatividade e portabilidade. RIA é a fusão de interatividade e multimídia na interface do usuário e funcionalidades de aplicações *desktop* com aplicações *Web* (PRECIADO et al., 2005). Atualmente muitas das novas aplicações para *Web* desenvolvidas fazem uso das tecnologias RIAs, em virtude das características apresentadas.

Com o intuito de atender a um cliente e também como forma de aprendizado de novas tecnologias, propõe-se neste trabalho o desenvolvimento de uma Aplicação Rica para Internet (RIA) para o gerenciamento de uma oficina mecânica, utilizando como base o padrão de desenvolvimento *Model View Controller* (MVC) (MACORATTI, 2013).

A arquitetura MVC fornece uma maneira de separar a funcionalidade envolvida na manutenção e na apresentação dos dados de uma aplicação. O padrão de arquitetura de software MVC consiste basicamente em separar a parte da lógica de negócio da parte de apresentação, juntamente com a camada de controle, sendo essa camada intermediária entre as outras duas, responsável por controlar e mapear as ações. Essas camadas funcionam de forma independente, o que permite o desenvolvimento, teste e manutenção isolada das partes, facilitando o seu desenvolvimento e a futura manutenção do sistema.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Desenvolver uma aplicação rica para Internet para gerenciamento de oficinas mecânicas.

1.2.2 Objetivos específicos

Para atingir o objetivo geral deste trabalho de conclusão de curso os seguintes objetivos específicos foram definidos:

- Estudo bibliográfico sobre o desenvolvimento baseado no padrão de arquitetura de software MVC.
- Estudo bibliográfico sobre Aplicações Ricas para Internet.
- Desenvolver a análise e o projeto;
- Realizar a implementação do sistema utilizando conceitos de aplicação Internet rica e padrão de arquitetura MVC.

1.3 JUSTIFICATIVA

Uma oficina mecânica funciona basicamente através da venda de serviços e algumas vezes também da venda de peças. Para o gerenciamento de uma oficina mecânica é necessário o controle dos serviços prestados, das peças vendidas, do estoque e dos contatos de clientes e fornecedores, entre outros. São vários os processos envolvidos e o controle destes processos de forma manual torna-se uma tarefa difícil de ser gerenciada.

Assim neste trabalho, a partir de uma demanda identificada, verificou-se a possibilidade e mesmo a necessidade de desenvolver um sistema informatizado para automatizar e melhorar os processos de uma oficina mecânica. E isso justifica a aplicabilidade do sistema.

A realização deste trabalho se baseia no uso de tecnologias para o desenvolvimento de aplicações denominadas Aplicações Ricas para Internet. RIAs permitem construir aplicações ricas com dados e conteúdo multimídia, alta interatividade, incrementando as capacidades oferecidas pelas aplicações *Web* tradicionais. O desenvolvimento desse sistema é uma oportunidade do acadêmico colocar em prática o aprendizado obtido no curso de graduação, assim como permite

o uso de novas tecnologias e metodologia de organização do software, isto é, como implementar uma aplicação utilizando três camadas, baseado no padrão MVC.

1.4 ORGANIZAÇÃO DO TRABALHO

Este texto está organizado em cinco capítulos, sendo que o primeiro apresenta a ideia do sistema com os objetivos e a justificativa.

O Capítulo 2 apresenta o referencial teórico e descreve as aplicações *Web* tradicionais, conceitos de sobre Aplicações Ricas para Internet e a arquitetura de software MVC.

O Capítulo 3 descreve as tecnologias e ferramentas utilizadas para o desenvolvimento deste trabalho.

No Capítulo 4 está o sistema desenvolvido, incluindo a apresentação do sistema, modelagem de telas, implementação do sistema, testes e como foi efetuada a implantação do sistema.

No Capítulo 5 são apresentadas as conclusões a respeito deste trabalho.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico do trabalho abrangendo aplicações *Web* tradicionais, aplicações Ricas para Internet, assim como uma breve comparação entre estas tecnologias. Também descreve a arquitetura MVC de desenvolvimento de software.

2.1 APLICAÇÕES WEB TRADICIONAIS

A *Web* teve um grande aumento de popularidade em 1993, quando o navegador Mosaic foi lançado, e continuou a crescer rapidamente durante a década de 1990. Em 2003 estava ocorrendo uma mudança perceptível na maneira de usar a *Web* e desenvolver aplicações baseadas na *Web* (DEITEL, 2008).

Para Aniceto (2009, p.2) aplicação *Web* pode ser definida como: “sistema de informática projetado para a utilização de um navegador, na Internet ou em redes privadas e o seu desenvolvimento tem muito haver com a necessidade de simplificar a utilização e manutenção, mantendo o código fonte em um mesmo local, de onde é acessado pelos diferentes usuários”.

Uma Aplicação *Web* consiste em uma aplicação que é executada através de um navegador (*browser*) na Internet ou em redes privadas (Intranet). Uma aplicação *Web* é geralmente estruturada na arquitetura cliente-servidor. Nessa arquitetura o processamento das informações entre o lado cliente (usuário do sistema) e o servidor (unidade de processamento das informações) são separados, sendo estas duas estruturas interligadas por meio de rede de computadores.

Para Kennet e Jane Laudon (1998, p.54) “o servidor pode armazenar programas aplicativos e arquivos de dados e pode distribuir programas ou arquivos de dados para outros computadores da rede à medida que estes os solicitam”.

As instâncias clientes realizam requisições para o servidor da aplicação e aguardam o retorno da resposta. O servidor disponível para a aplicação pode aceitar as requisições, processá-las e retornar o resultado para o cliente. Este modelo de aplicação se caracteriza pela facilidade em ser atualizado e mantido, pois seus

códigos e bancos de dados ficam localizados em apenas um servidor.

Nas aplicações *Web* para que clientes e servidores se entendam, ambos devem utilizar o mesmo protocolo, isto é, as mesmas regras. O protocolo usado por ambos é o protocolo HTTP (*Hypertext Transfer Protocol*) (RFC2616, 1999).

A Figura 1 mostra as interações entre cliente e servidor em uma aplicação *Web* tradicional, como em um site com formulário de registro do usuário. O usuário preenche os campos no formulário e submete-o (etapa 1), o navegador gera uma solicitação ao servidor, que a recebe e processa (etapa 2), então o servidor gera e envia uma resposta contendo a página que será exibida no navegador do usuário (etapa 3) que carrega a nova página (etapa 4), nesse processo o cliente espera o servidor processar e recarregar a página inteira, enquanto espera, o usuário não pode interagir com a página. O processo inicia novamente quando o usuário interage com a página em outro formulário e o submete (etapa 5 a 8), esse comportamento das aplicações *Web* tradicionais de carregar a página inteira fez com que os usuários exigissem um comportamento responsivo encontrado nas aplicações *Desktop* cliente-servidor (DEITEL, 2008).

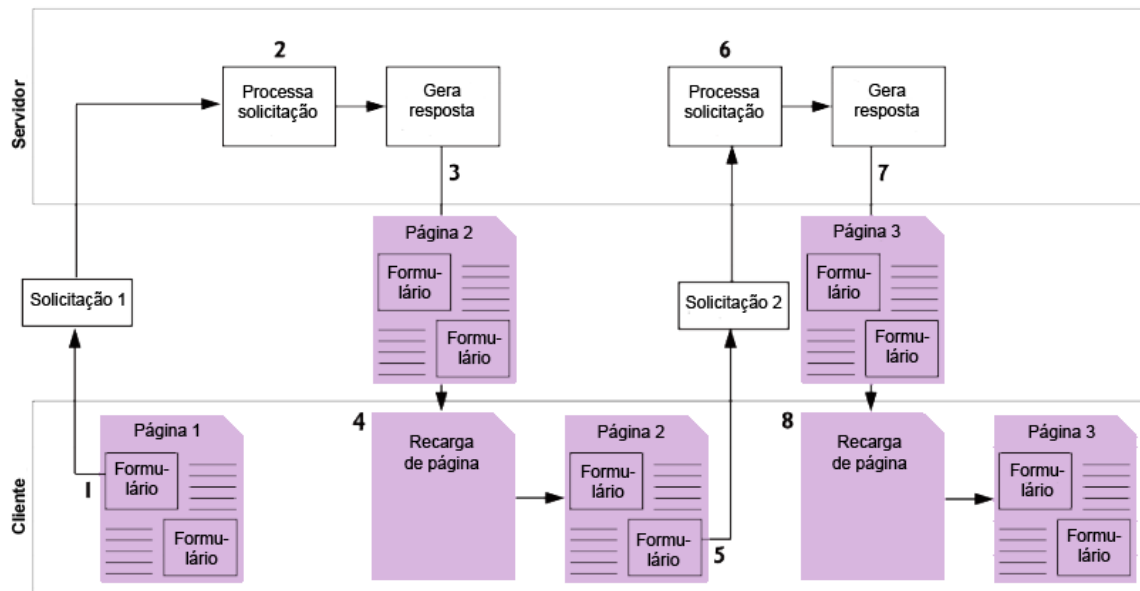


Figura 1 – Interações entre cliente e servidor
 Fonte: Adaptado de Deitel e Deitel (2008).

Com o passar do tempo as aplicações *Web* tradicionais não supriam mais as necessidades das pessoas e dos negócios, pois as primeiras aplicações para *Web* suportavam apenas HTML básico com funções simples, não sendo possível um retorno adequado ao usuário como em jogos, que necessitam trocas de cenários e

movimentações rápidas de personagens, portanto se faziam necessárias aplicações com aparência e sensação de uma aplicação Desktop (DEITEL, 2008). Assim, surgiram as aplicações ricas para Internet.

2.2 APLICAÇÕES RICAS PARA INTERNET (RIA)

De acordo com Paul e Harvey Deitel (2008, p.23) “Rich Internet Applications – RIA (Aplicações Ricas para Internet) são aplicações *Web* que oferecem a sensibilidade, recursos e funcionalidade ‘ricas’, que se aproximam das aplicações de Desktop”. As RIAs são aplicações mais interativas e mais ágeis do que aplicações *Web* tradicionais, tendo como características as facilidades de arrastar e soltar, animações multimídia, sem necessidade de recarregar a página e tempo de resposta curto.

2.2.1 Motivação para Aplicações RIA

Para suprir os problemas e as necessidades das aplicações *Web* tradicionais, foi proposta uma abordagem de aplicações com mais funcionalidades e complexidades, chamada de Aplicação Rica para Internet (RIA – *Rich Internet Application*), termo introduzido pela Macromedia em um artigo escrito por Jeremy Allaire em 2002, fazendo menção a unificação de aplicações *Desktop* com aplicações *Web* (BOZZON, 2006). Essas aplicações provêm interfaces sofisticadas para representar processos e dados complexos, minimizando a transferência de dados e transferindo as camadas de apresentação e interação do lado do servidor para o lado do cliente (BOZZON, 2006).

Duhl (2003) apresenta alguns pontos que as RIAs devem ser capazes de desempenhar para suprimir a simplicidade de aplicações *Web* tradicionais:

- Possuir um cliente que pode fazer utilização da conexão em qualquer momento e que seja portátil para qualquer aplicação;

- Ter um poder de execução que não comprometa seu funcionamento em conexões lentas e rápidas;
- Utilizar componentes de áudio, vídeo e texto simultaneamente sem emendas;
- Acessar diversas camadas de negócio, como .NET e Java, e armazenamento de dados;

A Figura 2 ilustra uma RIA, como sendo uma combinação das melhores funcionalidades de uma aplicação *Desktop*, de uma aplicação *Web* e interatividade, como recursos multimídia (comunicações). O desktop contribui com as interfaces gráficas interativas com o usuário, respostas mais rápidas das interfaces sem recarregamento de página e comportamentos como arrastar e soltar. A parte originada das aplicações *Web* inclui funcionamento sem depender de plataformas, apenas de um único cliente (navegador), a possível utilização de download progressivo para transferência de dados e o padrão adotado na Internet.



Figura 2 – RIA combina o melhor do *Desktop*, da *Web* e de *Comunicações*
Fonte: Adaptado de DUHL (2003).

2.2.2 Características de RIA

As RIAs têm características similares as aplicações *Desktop*, como aparência, comportamento e usabilidade. O Quadro 1 mostra uma comparação de características entre aplicações *Desktop* incluindo cliente-servidor, aplicações *Web* convencionais (HTML + HTTP) com aplicações que ressaltam o potencial das RIAs, que dependem de um modelo de distribuição *Web*, oferecem interfaces melhoradas e reduzem a sobrecarga de comunicação. Esses benefícios são relevantes para uma variedade de tarefas, como armazenamento de dados, filtragem e renderizações complexas. Normalmente, uma RIA é carregada pelo cliente com alguns dados iniciais, então ele gerencia a renderização de dados e o processamento de eventos, comunicando com o servidor somente quando o usuário necessita mais informações ou envia dados (BOZZON, 2006).

Característica	Cliente/Servidor <i>Desktop</i>	Web	RIA
Cliente universal (browser)	Não	Sim	Sim
Instalação no cliente	Complexa	Simple	Simple
Capacidades de interação	Rica	Limitada	Rica
Lógica de negócios do lado do servidor	Sim	Sim	Sim
Lógica de negócios do lado do cliente	Sim	Limitada	Sim
Necessidade de recarregar a página toda	Não	Sim	Não
<i>Round-trips</i> (ida e volta) de servidor frequentes	Não	Sim	Não
Comunicação servidor cliente	Sim	Não	Sim
Funcionamento quando desconectado de rede	Sim	Não	Sim

Quadro 1 - Comparação entre *Desktop* incluindo cliente e servidor, aplicações *Web* e RIA
Fonte: Bozzon (2006).

Algumas características que podem ser consideradas particulares de RIA são (BOZZON, 2006):

- Reduzir a comunicação com o servidor ao mínimo e suportar processamento no lado do cliente;
- Vários níveis de persistência para os dados que compõem a aplicação, tanto no lado do cliente quanto no servidor;

- O processamento de dados pode ocorrer em ambos os lados, cliente e servidor;
- A utilização pode ocorrer em modos on-line e off-line.

2.2.3 Funcionamento de RIA

Dois atributos são essenciais em uma RIA, o desempenho e a interface rica. O desempenho é obtido através do Ajax (*Asynchronous JavaScript and XML*) (GARRET, 2005), que possibilita atualizar partes individuais de uma página *Web* sem necessidade de recarregar a página inteira, isso possibilita que os usuários continuem interagindo com a página enquanto o servidor processa as requisições, criando assim uma interface visual mais sensível, reduzindo a espera do usuário. A interface visual rica é obtida através de ferramentas e *frameworks* RIA, como Adobe Flex (ADOBE, 2013a), o qual é utilizado nesse trabalho, que oferecem controles e funcionalidades poderosas e fáceis de usar para enriquecer as aplicações *Web* como ocorre com interfaces de *Desktop*.

Para a comunicação as aplicações Ajax adicionam uma camada entre o cliente e o servidor para gerenciar a comunicação entre os mesmos. A Figura 3 mostra as interações entre cliente, camada adicional e servidor, o cliente cria um objeto XMLHttpRequest para gerenciar uma solicitação (etapa 1) a partir da interação entre o usuário e a página, o objeto XMLHttpRequest envia a solicitação ao servidor (etapa 2) e espera a resposta. Como as solicitações são assíncronas, o usuário continua interagindo com a página enquanto o servidor processa a solicitação anterior, novas interações entre o usuário e a página geram outra solicitação ao servidor (etapas 3 e 4). Quando o servidor responde a primeira solicitação (etapa 5) o objeto XMLHttpRequest aciona uma função de *callback* no lado do cliente que processa os dados retornados e faz atualizações parciais da página (etapa 6) para exibir os dados sem necessidade de recarregar toda a página. Ao mesmo tempo, o servidor pode responder a segunda solicitação (etapa 7) e o cliente faz outra atualização de página parcial (etapa 8). Com as atualizações de página parciais as aplicações *Web* adquirem a agilidade encontrada em aplicações *Desktop*.

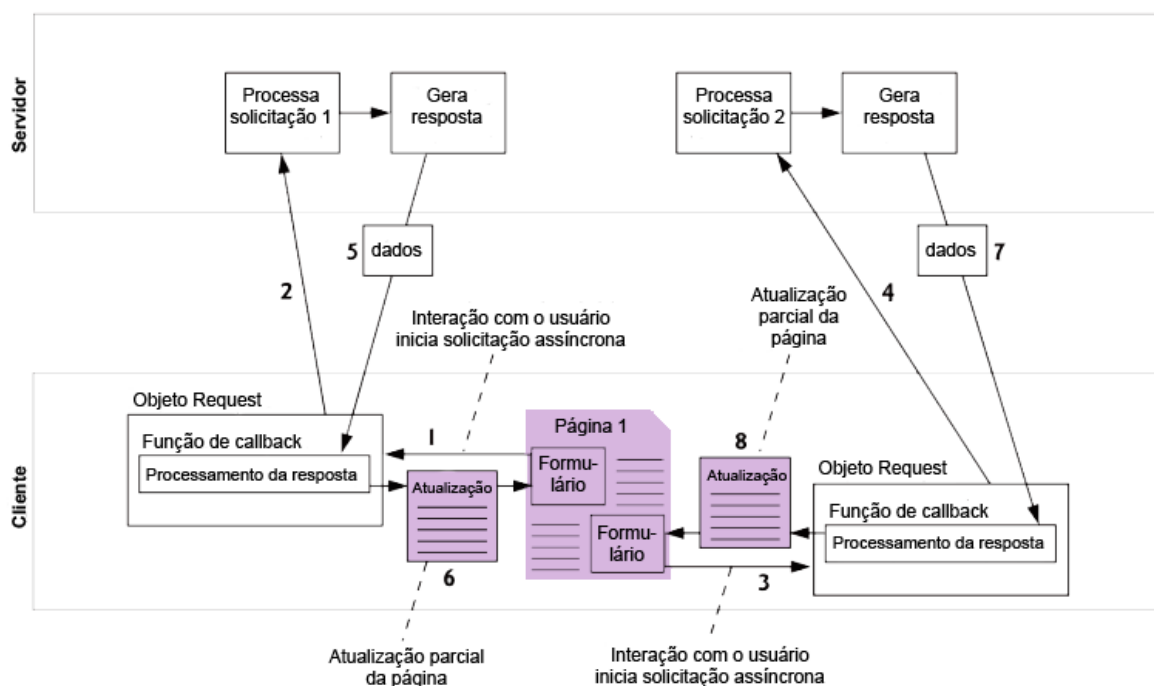


Figura 3 – Interações entre cliente, camada adicional e servidor
 Fonte: Adaptado de Deitel e Deitel (2008).

2.2.4 Tecnologias RIA existentes

Existem diversas tecnologias e ferramentas disponíveis para o desenvolvimento de RIA, entre elas destacam-se:

- Flex (ADOBE, 2013a) – é um *framework* RIA da Adobe que possibilita criar aplicações ricas em multimídia, para múltiplas plataformas e escaláveis, podendo essas aplicações serem distribuídas pela Internet.
- JavaFX (ORACLE, 2013a) – é um produto da Sun Microsystems. Consiste em uma linguagem de scripts (JavaFX Script) e um sistema para dispositivos móveis (JavaFX Mobile).
- JavaServer Faces* (ORACLE, 2013d) – é um *framework* de aplicação *Web* baseado em Java, separa os elementos de projeto da lógica de negócio e provê um conjunto de componentes de interface de usuário que simplificam o desenvolvimento de RIAs.
- Google Web Toolkit* (GOOGLE, 2013) – é uma ferramenta do Google que permite criar aplicações ricas, transformando código Java em JavaScript otimizado.

Neste trabalho a tecnologia Flex foi usada como objeto de estudo, pois trata-se de uma amplamente utilizada no mercado (ADOBE, 2013d), também porque o autor deste trabalho possui experiência nesta tecnologia e atende aos requisitos necessários para o desenvolvimento da aplicação.

2.3 ARQUITETURA DE SOFTWARE MVC

MVC também chamado de Model View Controller é um padrão de projeto. Para Gamma et al. (2006, p.20), um padrão de projeto “são descrições de objetos e classes comunicantes que precisam ser personalizadas para resolver um problema geral de projeto num contexto particular”.

De acordo com Gamma et al. (2006), a abordagem MVC é composta por três tipos de objetos: o Modelo é o objeto de aplicação, a Visão é a apresentação na tela e o Controlador é o que define a maneira como a interface do usuário reage às entradas do mesmo.

A abordagem MVC separa visão e modelos pelo estabelecimento de um protocolo do tipo subscrição/notificação (*subscribe/notify*) entre eles. Assim sendo, uma visão deve garantir que seu aspecto reflita no estado do modelo, quando os dados do modelo mudam, o modelo notifica as visões que dependem dele. Em resposta cada visão tem a possibilidade de atualizar-se. Essa abordagem permite ligar várias visões a um modelo para fornecer diferentes apresentações, conforme ilustrado na Figura 4 (GAMMA, et al, 2006).

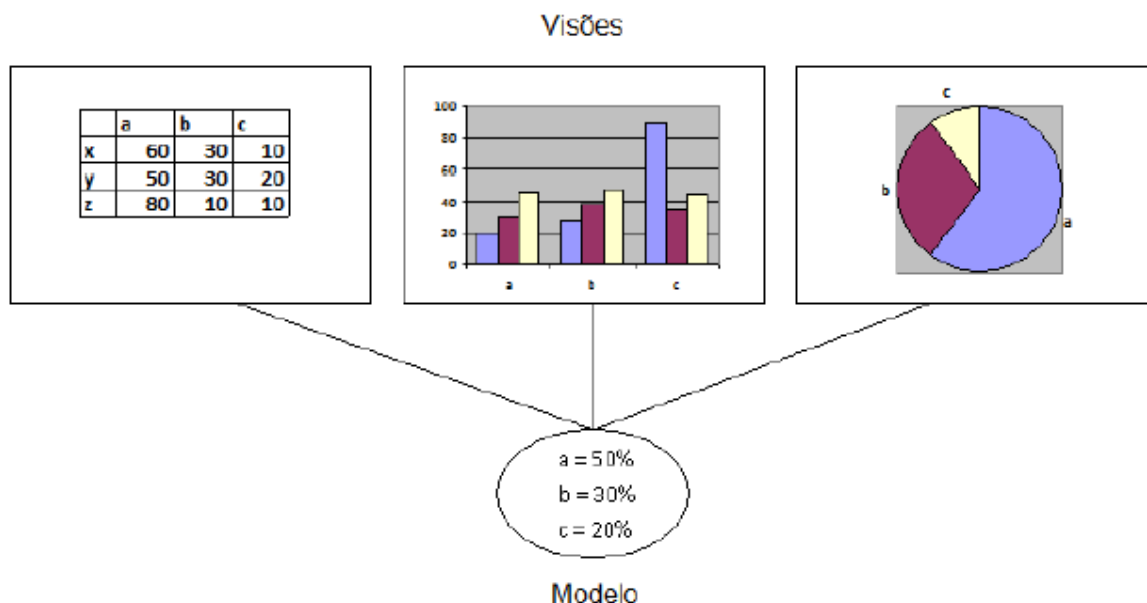


Figura 4 – Visões ligadas a um modelo
Fonte: Adaptado de Gamma et al. (2006).

Segundo MACORATTI (2013) as vantagens da arquitetura de software MVC são:

- a) Como o modelo MVC gerencia múltiplos visualizadores usando o mesmo modelo é fácil manter, testar e atualizar sistemas múltiplos;
 - b) É muito simples incluir novos clientes apenas incluindo seus visualizadores e controles;
 - c) Torna a aplicação escalável;
 - d) É possível ter desenvolvimento em paralelo para o modelo, visualizador e controle, pois são independentes.
- e) Ainda segundo MACORATTI (2013) as desvantagens do modelo MVC são:
- f) Requer uma quantidade maior de tempo para analisar e modelar o sistema;
 - g) Requer pessoal especializado;
 - h) Não é aconselhável para pequenas aplicações.

3 MATERIAIS E MÉTODO

Este capítulo apresenta as ferramentas, as tecnologias e o método utilizado para o desenvolvimento do sistema.

3.1 MATERIAIS

Neste trabalho, foi utilizado Apache Flex (Apache, 2013a) para camada de visão. Para a camada de controle, foi utilizado a linguagem Java para as regras de negócio, o Framework BlazeDS (ADOBE, 2013a) para a comunicação entre o Adobe Flex e Java (ORACLE, 2013b). Para a camada de modelo foi usado o gerenciador de banco de dados MySQL (ORACLE, 2013c) e o Framework Hibernate (REDHAT, 2013) para auxiliar na persistência de dados. Para a geração de relatórios foi utilizado o BIRT (ECLIPSE, 2013b).

O Quadro 2 apresenta um resumo das ferramentas utilizadas, apresentando o nome das mesmas, a versão utilizada, o site onde podem ser obtidas e uma breve descrição.

Tecnologia /Ferramenta	Versão	Site	Descrição
Java	6.0	http://www.oracle.com	Linguagem de programação
Adobe Flex	3.6	http://www.adobe.com	Desenvolvimento das interfaces visuais do sistema.
Adobe Flash Builder	4.6	http://www.adobe.com	Ambiente de desenvolvimento para Flex.
Eclipse IDE	3.7	http://www.eclipse.org	Ambiente de desenvolvimento para Java.
Apache Tomcat	6.0	http://www.tomcat.apache.org	Servidor <i>Web</i> Java
Adobe BlazeDS	4.0	http://www.adobe.com	Integração Java e Flex
BIRT	3.7	http://www.eclipse.org/birt/phoenix/	Relatório em Java.

MySQL	5.5	http://www.mysql.com	Gerenciamento de banco de dados
-------	-----	---	---------------------------------

Quadro 2 - Lista de ferramentas e tecnologias utilizadas

3.1.1 Linguagem de Programação Java

A tecnologia Java (ORACLE, 2013b) foi criada como uma ferramenta de programação para computadores, foi parte de um pequeno trabalho anônimo e secreto chamado “*The Green Project*” da Sun Microsystems em 1991. Eles estavam tentando antever e planejar a tendência na computação e de dispositivos controlados digitalmente. O primeiro programa que a equipe *Green Team* desenvolveu foi um controlador portátil para sistemas de entretenimento doméstico voltado para o setor de televisão digital. Mas esse segmento do mercado não estava preparado para receber tal poder de desenvolvimento da linguagem. Como a Internet estava pronta para essa tecnologia e bem a tempo para sua apresentação pública em 1995, a equipe pode anunciar que o navegador Netscape Navigator passaria a incorporar a tecnologia Java. Sua versatilidade, eficiência, portabilidade de plataforma e segurança fazem dela a tecnologia ideal para computação em rede (SERSON, 2007).

Atualmente os direitos da linguagem Java são de propriedade da empresa Oracle (ORACLE, 2013b). Hoje em dia a tecnologia Java encontra-se em redes e dispositivos que vão desde a Internet e supercomputadores científicos a *laptops* e telefone celulares, simuladores de mercados financeiros a dispositivos para jogos e cartões de crédito.

Uma razão para a popularidade do Java é a sua independência de plataforma. Isto significa que o mesmo programa compilado pode executar em qualquer computador. Essa independência torna Java diferente da maioria das outras linguagens de programação, que exigem diferentes compiladores para diferentes sistemas operacionais (HUBBARD, 2004).

Além de sua conveniência e eficiência, a independência de plataforma possui outra grande vantagem para programas que são executados em uma rede. Um programa Java compilado pode ser armazenado em um único servidor, e as

máquinas clientes de qualquer tipo podem facilmente fazer o *download* e executar o programa. Java faz isso através da compilação de seu código fonte para uma linguagem genérica *bytecode*, que é executada pela máquina cliente usando um programa denominado de *Java Virtual Machine* (JVM). Do mesmo modo que o código fonte, *bytecodes* são independentes do tipo de sistema de computação. O mesmo arquivo de *bytecode* pode ser usado por qualquer computador (HUBBARD, 2004).

Outras razões para a popularidade do Java incluem seu suporte para a programação orientada a objetos e sua vasta coleção de bibliotecas de classes.

3.1.2 Apache Flex

O *framework* Apache Flex foi desenvolvido para permitir os desenvolvedores de sistemas criarem com rapidez aplicações RIA para diversas plataformas e conteúdos usando a estrutura Flex de código aberto. Com suporte para codificação com alta produtividade, depuração e *design* visual, também oferece ferramentas de teste que aceleram o desenvolvimento, resultando em aplicativos de desempenho superior (APACHE, 2013a).

O Flex possibilita o desenvolvimento de aplicações *Web*, utilizando o Flash Player para poder ser renderizado dentro do navegador. Um sistema é composto por botões, caixas de texto, tabelas e gráficos, devidamente organizados de forma que o usuário possa usar o sistema sem dificuldades, com boa usabilidade e com funcionamento rápido e conciso. O Flex permite isso dentro do navegador de forma simples, algo tecnicamente difícil de fazer com HTML, Java Script e CSS (*Cascading Style Sheets*) (SCHMITZ, 2010).

O Flex é basicamente usado para a criação dos leiautes, operações como salvar informações no banco de dados, enviar *e-mail*, gerar arquivos pdf, etc., são realizadas no servidor, utilizando a linguagem de preferência: PHP (*Hypertext Preprocessor*), Java, .Net, etc. Flex utiliza MXML (*Minimal XML*), que é uma linguagem de marcação de texto, semelhante ao HTML, também utiliza ActionScript, que é uma linguagem de programação orientada a objetos (SCHMITZ, 2010).

As aplicações para *Web* feitas em Flex possuem a necessidade de ter instalado o Adobe Flash Player (ADOBE, 2013b) no navegador do usuário, para executar a aplicação, sendo essa uma desvantagem na utilização do *framework*.

O Flex foi desenvolvido por a Adobe (ADOBE, 2013c) e em junho de 2012 foi doado a Apache (APACHE, 2013a), que ficou responsável pela manutenção do projeto e lançamento de novas versões.

3.1.3 Eclipse IDE

Eclipse é uma IDE (*Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento) de código aberto utilizado para a construção de programas de computador (ECLIPSE, 2013a). O projeto foi iniciado pela IBM, que desenvolveu a primeira versão do produto e o doou como software livre a comunidade. Possui como características principais a orientação ao desenvolvimento baseado em *plug-ins* e o suporte ao desenvolvedor, com centenas de *plug-ins* que procuram atender as diferentes necessidades de diferentes programadores (SERSON, 2007).

Para o desenvolvimento do sistema de gerenciamento de oficina mecânica foi utilizado o Eclipse para desenvolvedores Java versão Indigo. O projeto criado foi do tipo *Dynamic Web Project*, que tem o objetivo de desenvolvimento de uma aplicação *Web*. Esse tipo de projeto já vem com configurações básicas estabelecidas, facilitando assim o trabalho do desenvolvedor, que não perde tempo com configurações, como estrutura de pastas e servidor Java.

A Figura 5 mostra o Eclipse sobre a perspectiva Java com o projeto do tipo *Dynamic Web Project* aberto.

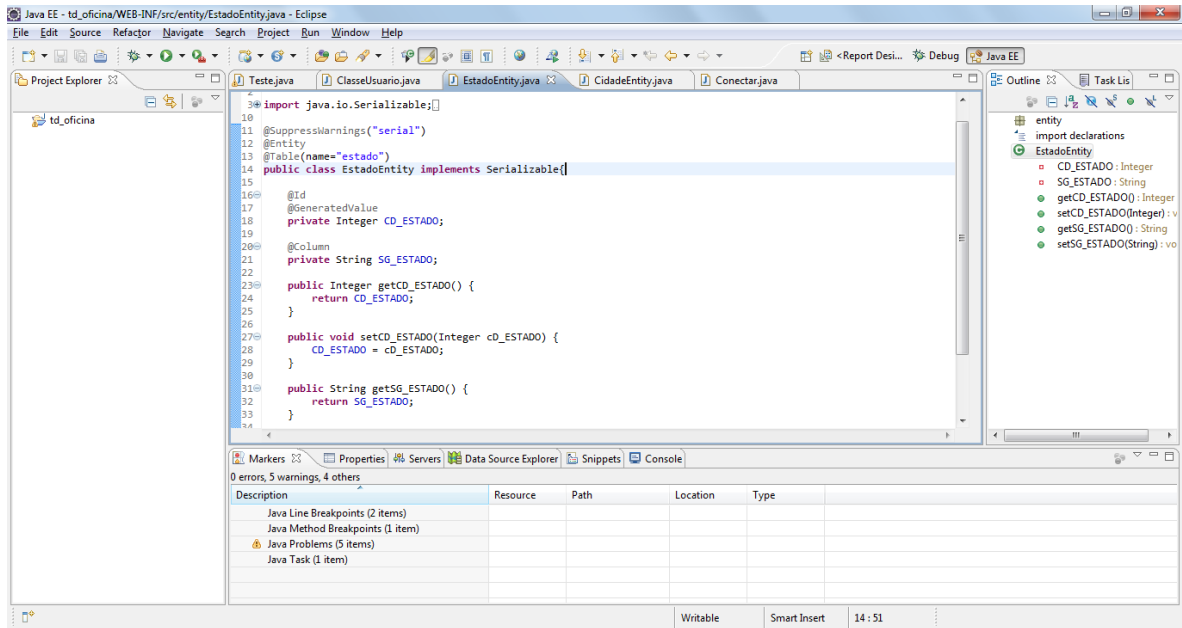


Figura 5 - Eclipse IDE sob a perspectiva Java

3.1.4 Flash Builder

Para o desenvolvimento do sistema de gerenciamento de oficina mecânica foi utilizado o Flash Builder 4, que é uma IDE de desenvolvimento em Flex baseada no Eclipse IDE, com recursos adicionais como criação de projetos em Flex, conexão com servidor de dados, perfil de desempenho da aplicação, etc., a Figura 6 mostra o Flash Builder 4 utilizando a perspectiva “Flash”.

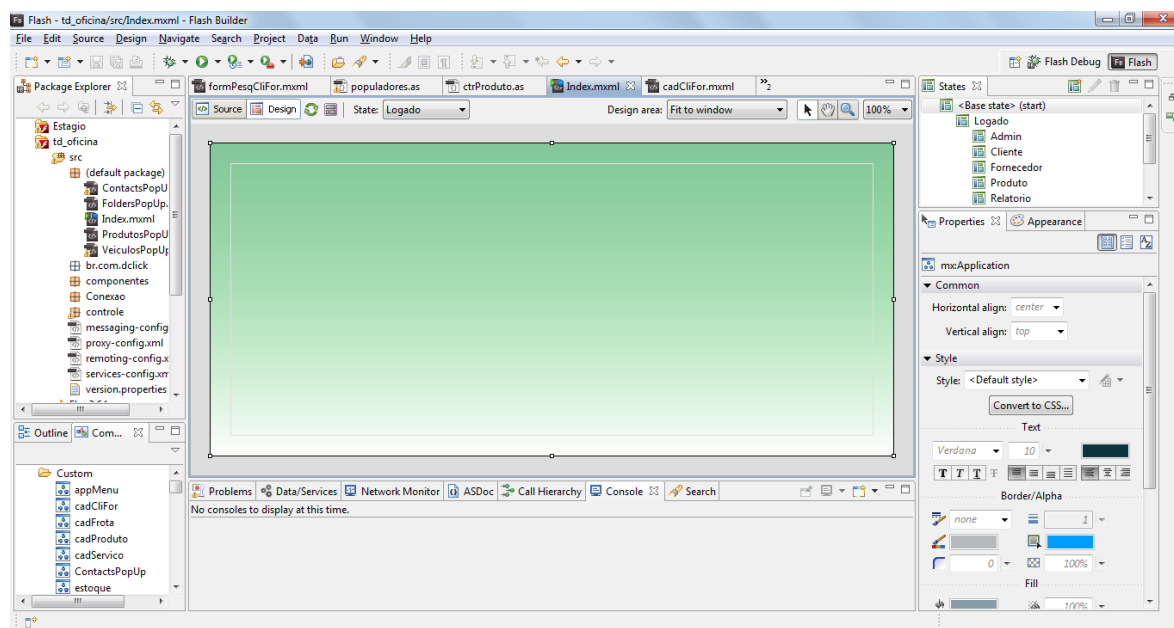


Figura 6 - Flash Builder 4 na perspectiva Flash

3.1.5 Apache Tomcat

O Tomcat (APACHE, 2013b) é um contêiner de Java Servlets que interpreta aplicações desenvolvidas em Java para *Web*. O Tomcat possui algumas características de um servidor de aplicação, mas como não preenche todos os requisitos necessários não pode ser considerado um servidor de aplicação, por exemplo, não tem suporte a EJB (*Enterprise JavaBean*). É distribuído como software livre sobre a licença Apache versão 2 (APACHE, 2013c) e foi desenvolvido originalmente pela Apache Software Foundation. O Tomcat abrange parte da especificação J2EE com tecnologias como Java Servlet e JSP (*Java Server Pages*) e tecnologias de apoio relacionadas como segurança, JNDI (*Java Naming and Directory Interface*) Resources e JDBC (*Java Database Connectivity*) DataSources.

O Tomcat também pode funcionar como um servidor *Web*, provendo um servidor *Web* HTTP puramente em Java, ou pode atuar integrado a um servidor *Web* dedicado, como o IIS (*Internet Information Services*). O Tomcat inclui ferramentas para configuração e gerenciamento, o que também pode ser feito editando seus arquivos de configuração no formato XML.

3.1.6 Adobe BlazeDS

O Adobe BlazeDS (ADOBE, 2013a) é uma tecnologia que possibilita o acionamento remoto e o envio de mensagens na comunicação entre a plataforma Adobe Flex e Java. É um programa gratuito e com código aberto distribuído sob a licença GNU *Lesser General Public License Version 3* (GNU, 2013) é executado em servidores de aplicação Java e possui uma série de recursos, entre eles estão:

- a) *Remoting Service* – Permite a aplicação Flex invocar diretamente métodos implementados no servidor *Web Java*;
- b) *Message Service* – Possibilita o envio de mensagens de modo assíncrono;
- c) *Proxy Service* – Permite a aplicação Flex acessar um serviço em um domínio diferente do seu.

O BlazeDS utiliza o protocolo AMF (*Action Message Format*) para o transporte de mensagens, que serializa os objetos em um formato binário compacto. Assim elimina a camada de abstração de dados, utilizada na comunicação baseada em XML, conseguindo um melhor desempenho no lado do servidor e no lado do cliente, além de resultar em uma comunicação mais eficiente entre cliente e servidor (WARD, 2013).

A Figura 7 ilustra a comunicação utilizando protocolos baseados em XML e AMF para enviar dados para a aplicação cliente. Na comunicação com XML muitos dos dados que estão sendo transferidos não são absolutamente necessários, como marcações de início e marcações de fim utilizadas para identificar os tipos de dados que são enviados. Quando os dados são recebidos na aplicação cliente é necessário convertê-los de acordo com as marcações. Com o protocolo AMF isso não ocorre, porque o protocolo AMF trabalha com tipos de dados nativos do Flash Player, assim a serialização e desserialização de dados são transparentes ao desenvolvedor (FIEL, 2013).

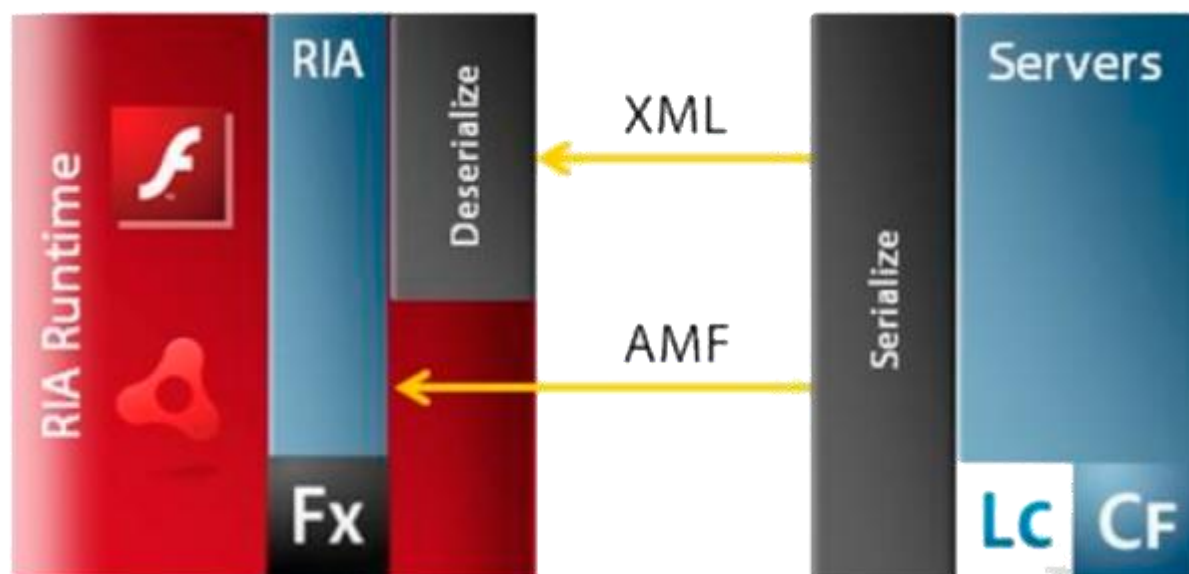


Figura 7 – Comparativo do protocolo AMF e XML
 Fonte: Adaptado de WARD (2013).

3.1.7 BIRT

O BIRT (*Business Intelligence and Reporting Tools*) (ECLIPSE, 2013b) é uma ferramenta para desenvolver relatórios para aplicações *Web*, baseadas em Java, possui código fonte aberto. O BIRT tem dois principais componentes, um componente para desenhar relatórios no Eclipse IDE (ECLIPSE, 2013a) e outro componente para executar os relatórios no servidor de aplicação. Permite a criação de gráficos de forma simples, acesso a diferentes fontes de dados, como JDBC (*Java Database Connectivity*), XML, *Web Services* e *scripts* de dados.

Os relatórios podem ser gerados em diferentes formatos, como PDF (*Portable Document Format*), HTML, XML. A estrutura do relatório é definida em um arquivo XML, mas com o componente de desenho de relatório não é necessário editar esse arquivo XML. É possível definir os elementos do relatório arrastando e soltando o elemento da palheta de elementos para dentro do relatório, assim é possível definir texto estáticos, imagens, linhas, tabelas, listas, gráficos e também definir campos dinâmicos que serão preenchidos a partir de uma base de dados.

A Figura 8 mostra o componente de desenho de relatório do BIRT dentro do Eclipse IDE.

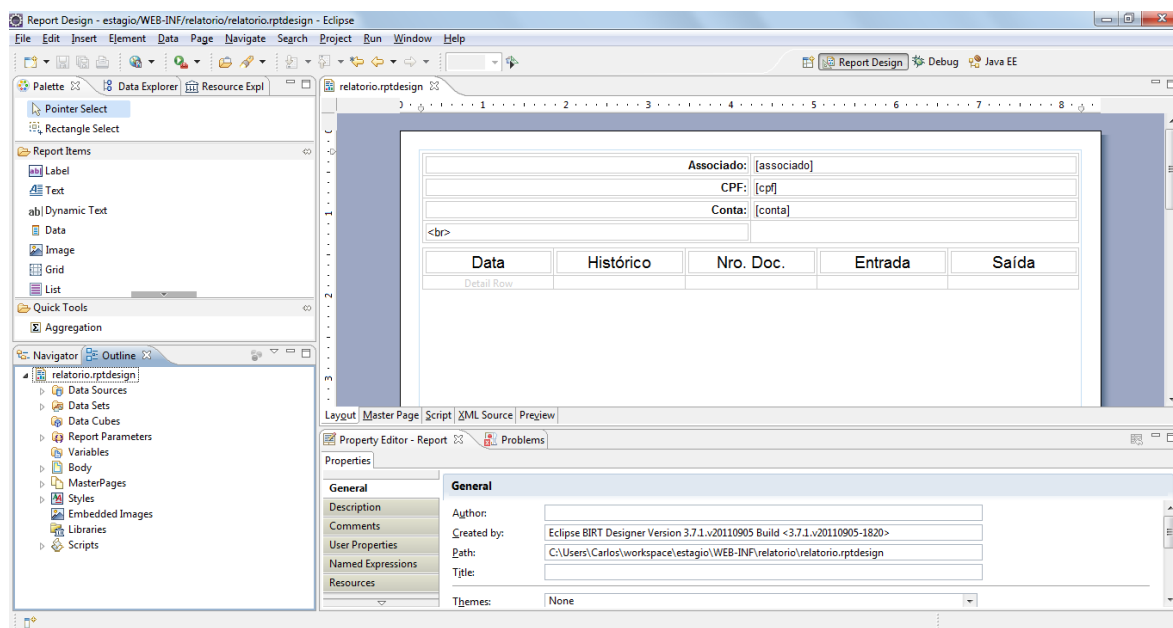


Figura 8 - Componente de criação de relatório do BIRT

3.1.8 MySQL

O MySQL (ORACLE, 2013c) é um sistema de gerenciamento de banco de dados que utiliza a linguagem SQL (*Structured Query Language*).

Entre as principais características destacam-se:

- a) Escalabilidade: oferece o máximo em termos de escalabilidade, tem a capacidade de rodar em aplicações embarcadas com espaço de apenas um *megabyte* para executar *data warehouses* de *terabytes* de informações.
- b) Flexibilidade: suporta diversas plataformas como Linux, UNIX e Windows.
- c) Alto Desempenho: a arquitetura do motor de armazenamento permite configurações específicas para cada aplicação resultando em alto desempenho.
- d) Robusto Suporte Transacional: as características incluem suporte completo a transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), ilimitado bloqueio em nível de linha, capacidade de transação distribuída e suporte a transações multi-versão, onde leitores nunca bloqueiam escritores e vice-versa.

- e) Código Aberto e Suporte 24 x 7: MySQL não é um projeto típico de código aberto, todo o sistema é de propriedade e apoiado pela Oracle com um custo único e um modelo de suporte disponível, oferece uma combinação única de sistema de código aberto e suporte confiável.

3.2 MÉTODO

O desenvolvimento do trabalho proposto foi dividido em algumas fases. Essas fases estão descritas a seguir:

- a) Levantamento de Requisitos – Foi analisado o funcionamento de uma oficina mecânica para identificar os requisitos e funcionalidades que se espera do sistema.
- b) Levantamento bibliográfico – O levantamento bibliográfico abrangeu a pesquisa de materiais sobre aplicações ricas para Internet e arquitetura MVC.
- c) Definição e estudo das tecnologias e ferramentas utilizadas – As tecnologias e ferramentas escolhidas para o desenvolvimento deste trabalho seguiram os seguintes critérios: popular, que possam atender aos requisitos levantados e, se possível, com licenças livres e/ou código aberto. O Quadro 2 apresenta as tecnologias e ferramentas utilizadas neste trabalho.
- d) Análise e projeto – Os requisitos auxiliaram a definir os casos de uso, o diagrama de banco de dados, as tabelas e respectivos campos do banco de dados e os relatórios que serão emitidos pelo sistema.
- e) Codificação – Foi definida ao decorrer do tempo uma estrutura hierárquica de pacotes e classes com o objetivo de deixar o projeto em desenvolvimento organizado e de acordo com a arquitetura MVC.
- f) Testes – Paralelamente ao desenvolvimento, serão feitos testes informais com o objetivo de garantir que o sistema estará funcional, atendendo as regras de negócios levantadas nos requisitos do sistema.
- g) Implantação – O sistema foi implantando em um servidor na empresa de um cliente.

4 RESULTADOS

Este capítulo apresenta o resultado da realização deste trabalho que é a implementação de um sistema para gerenciamento de oficina mecânica. É apresentada a descrição do sistema com suas funcionalidades, levantamento de requisitos, diagrama de caso de uso, diagrama de classes e a implementação do sistema com os resultados obtidos.

4.1 APRESENTAÇÃO DO SISTEMA

O sistema desenvolvido tem como finalidade informatizar o gerenciamento de uma oficina mecânica. O sistema provê o controle dos clientes e fornecedores, de peças vendidas e seu respectivo estoque, de veículos reparados e gerenciamento dos serviços prestados. Também possui controle de acesso ao sistema, em que somente um usuário previamente cadastrado e de possui senha pode efetuar acesso ao sistema.

Este sistema é destinado a um cliente do acadêmico, que antes da implantação do sistema na empresa, gerenciava todos os processos de forma manual ou com o auxílio de uma planilha eletrônica.

4.2 REQUISITOS

Para o desenvolvimento deste sistema foi analisado e levantado os requisitos necessários a fim de identificar as funcionalidades que o sistema deve possuir para atender as necessidades dos usuários e quais são as restrições que existem sobre essas funcionalidades.

O levantamento de requisitos está separado em dois tipos de requisitos que são os requisitos funcionais (RF) e requisitos não funcionais (RNF). Os requisitos

funcionais segundo Bezerra (2007 p.23) “definem as funcionalidades do sistema”, enquanto os requisitos não funcionais “declaram as características de qualidade que o sistema deve possuir e que estão relacionadas às suas funcionalidades”.

No Quadro 3 estão relacionados os requisitos funcionais do sistema e no Quadro 4 estão relacionados os requisitos não funcionais.

Requisito	Descrição
RF1	O sistema deve permitir a inclusão, alteração e exclusão de clientes.
RF2	O sistema deve permitir a inclusão, alteração e exclusão de fornecedores.
RF3	O sistema deve permitir a inclusão, alteração e exclusão de produtos.
RF4	O sistema deve permitir o controle de estoque de produtos
RF5	O sistema deve permitir o cadastro de veículos.
RF6	O sistema deve permitir o controle de serviços prestados.
RF7	O sistema deve possuir sistema de controle de acesso.

Quadro 3 – Requisitos funcionais

Requisito	Descrição
RNF1	O sistema deve ser executado independentemente de sistema operacional.
RNF2	O sistema deve rodar em qualquer navegador com o Flash Player instalado.

Quadro 4 – Requisitos não funcionais

4.3 DIAGRAMA DE CASOS DE USO

A Figura 9 apresenta o Diagrama de Casos de Uso do sistema proposto, com base nos requisitos levantados anteriormente. Conforme mostrado na Figura 9 é apresentado apenas um ator, mas pode variar o número de atores, de acordo com o controle de acesso.

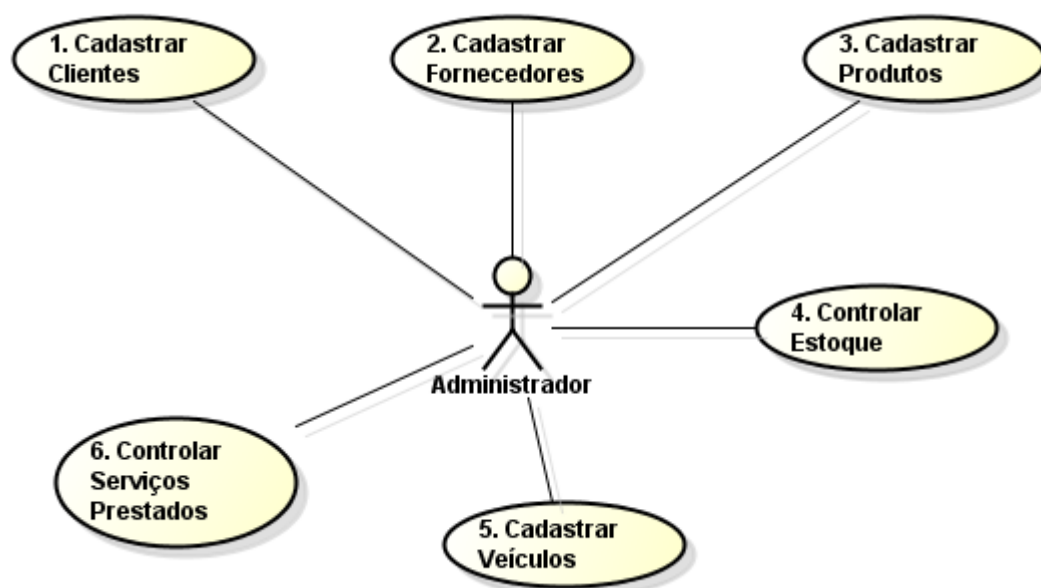


Figura 9 – Diagrama de casos de uso.

O Quadro 5 apresenta a descrição dos casos de uso apresentados no diagrama da Figura 9. Nesse quadro a sigla CSU significa caso de uso.

Caso de uso:	CSU01 - Cadastrar Clientes
Atores:	Administrador
Descrição:	O sistema permite incluir, alterar e excluir clientes.
Caso de uso:	CSU02 - Cadastrar Fornecedores
Atores:	Administrador
Descrição:	O sistema permite incluir, alterar e excluir fornecedores.
Caso de uso:	CSU03 – Cadastrar Produtos
Atores:	Administrador
Descrição:	O sistema permite incluir, alterar e excluir produtos.
Caso de uso:	CSU04 – Controlar Estoque
Atores:	Administrador
Descrição:	O sistema permite o controle de entrada e saída do estoque.
Caso de uso:	CSU05 – Cadastrar Veículos
Atores:	Administrador
Descrição:	O sistema permite incluir, alterar e excluir veículos.
Caso de uso:	CSU06 – Controlar Serviços Prestados
Atores:	Administrador
Descrição:	O sistema permite o controle de serviços prestados em veículos.

Quadro 5 - Descrição dos casos de uso

4.4 DIAGRAMA DE CLASSES

A Figura 10 apresenta as classes, com os atributos que são necessários no projeto, proporcionando uma visão geral de como o sistema deve funcionar.

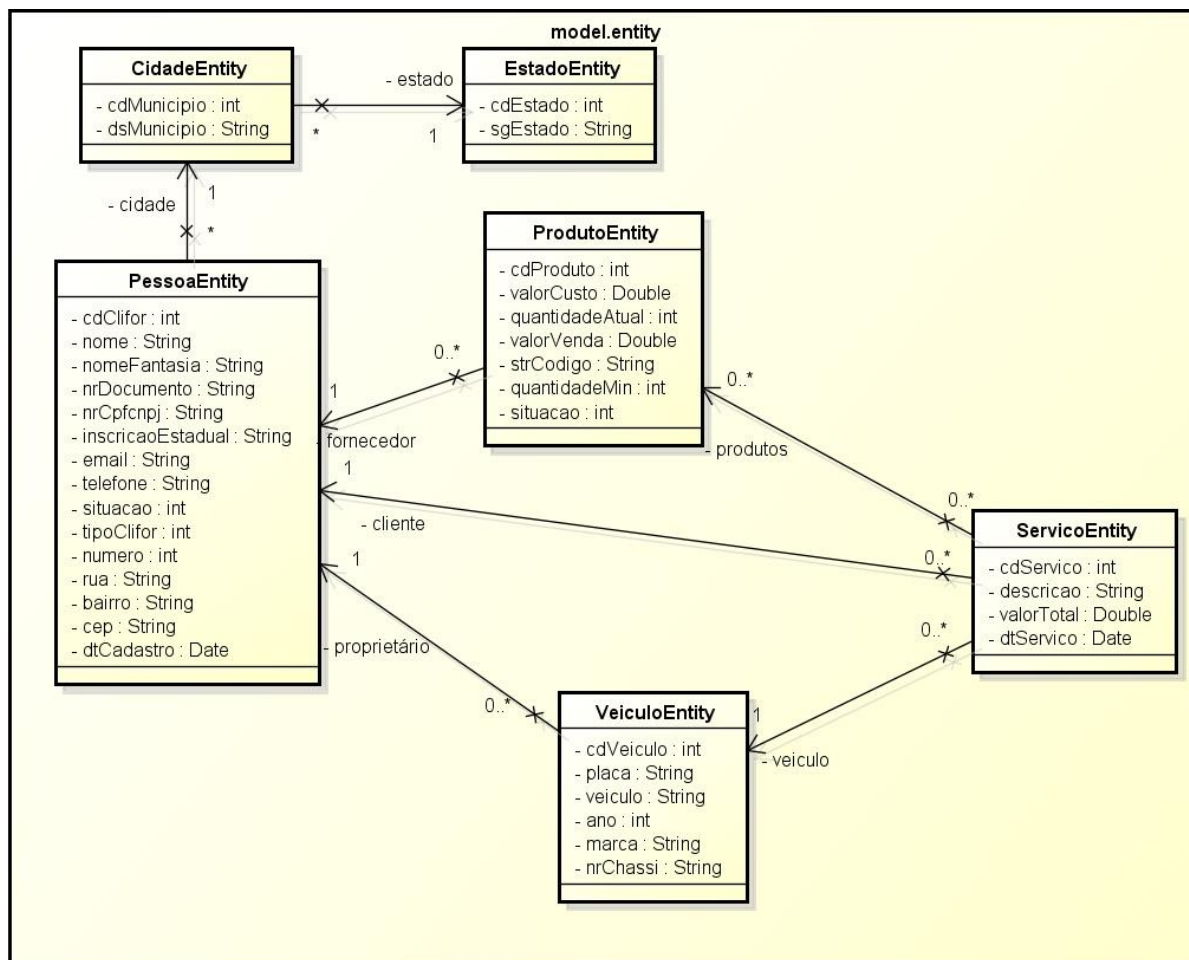


Figura 10 - Diagrama de classes

4.5 DIAGRAMA DE ENTIDADE RELACIONAMENTO

A Figura 11 apresenta o diagrama entidade relacionamento do sistema desenvolvido, no qual é possível visualizar a distribuição dos atributos nas entidades do banco de dados com seus relacionamentos entre as tabelas.

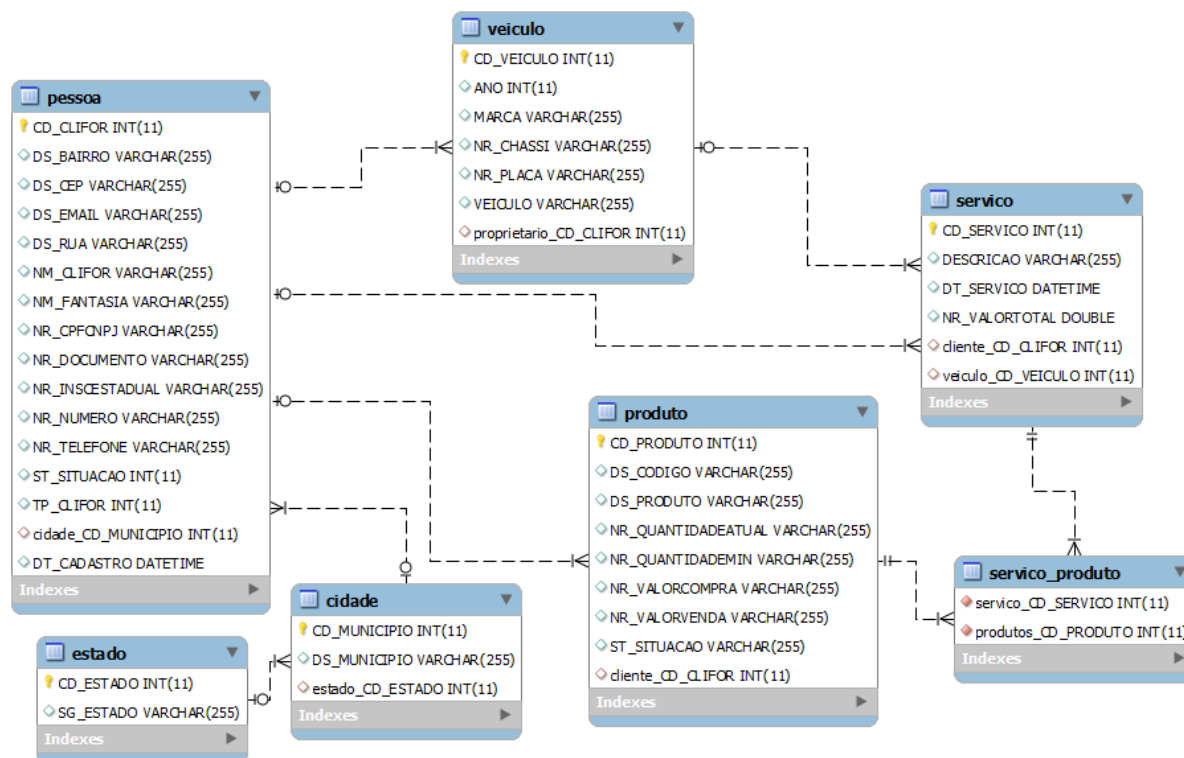


Figura 11 - Diagrama de entidade relacionamento

4.6 DESCRIÇÃO DO SISTEMA

A descrição do sistema apresenta a forma de uso por meio de suas telas, ou seja, a interação do usuário. Essa descrição é feita por meio das funcionalidades principais do sistema e é apresentada a seguir.

4.6.1 Autenticação no Sistema

Para acessar as funcionalidades do sistema o usuário precisa se autenticar com um usuário válido, para isso o sistema vem com um usuário administrador cadastrado, devem ser informados os valores “administrador” para o campo usuário e a para o campo senha o valor “%%m3c4n1c4”.

A Figura 12 apresenta a tela de autenticação do sistema, com os campos usuário e senha e o botão entrar.



A screenshot of a login form titled "Efetuar Login". It features two input fields: "Usuário:" and "Senha:". Below the fields is a button labeled "Entrar". The form is set against a light green background.

Figura 12 - Tela de autenticação do sistema

A autenticação tem validação de usuário e senha, caso os mesmos tenham sido informados incorretamente é apresentada uma mensagem ao usuário, conforme Figura 13.

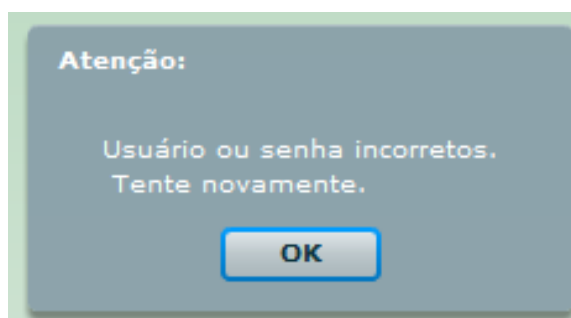


Figura 13 - Caixa de mensagem de validação

Após o usuário autenticar-se com sucesso no sistema é apresentada a tela inicial com o menu de opções conforme Figura 14, ao clicar em um item do menu é exibida a tela correspondente a função. As funcionalidades no menu são Clientes, Fornecedores, Produtos, Veículos e Serviços.

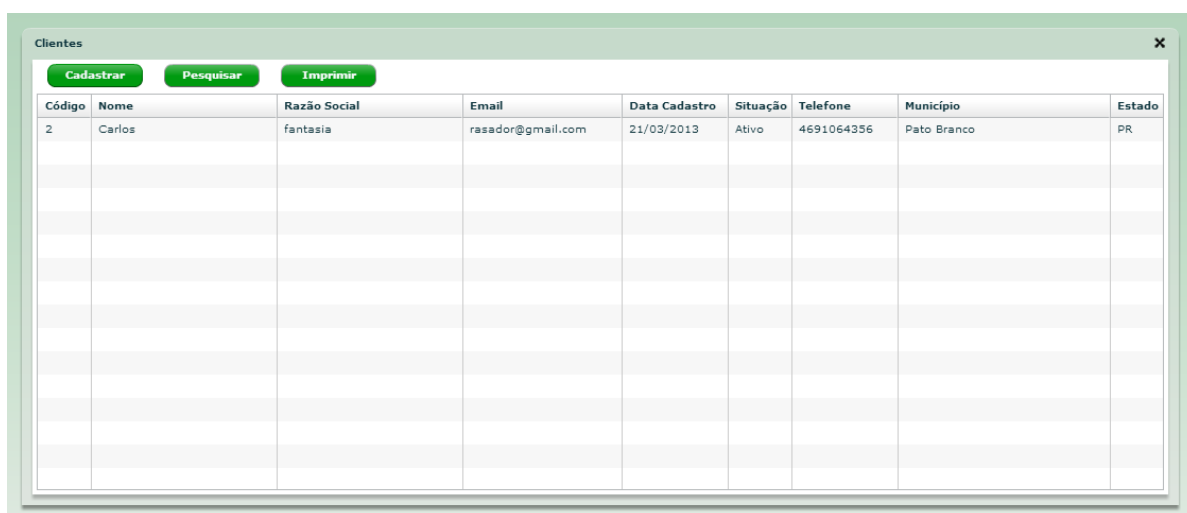


Figura 14 - Menu de opções do sistema

4.6.2 Cadastro de Clientes

As telas de cadastros do sistema seguem um padrão de leiaute, com os botões de funcionalidades do cadastro sendo exibidos no topo da tela e a lista de objetos sendo apresentada logo abaixo, ao abrir um cadastro a lista já exibe os registros existentes na base de dados.

Ao acessar o menu clientes é apresentada a tela do cadastro de clientes, conforme Figura 15, com a lista de clientes cadastrados e com os botões cadastrar, pesquisar e imprimir. A lista contém os campos código do cliente, nome do cliente, razão social, e-mail, data de cadastro do cliente, situação, podendo ser ativo ou inativo, telefone, município e sigla do estado. É possível editar um registro clicando duas vezes em cima do mesmo na lista.

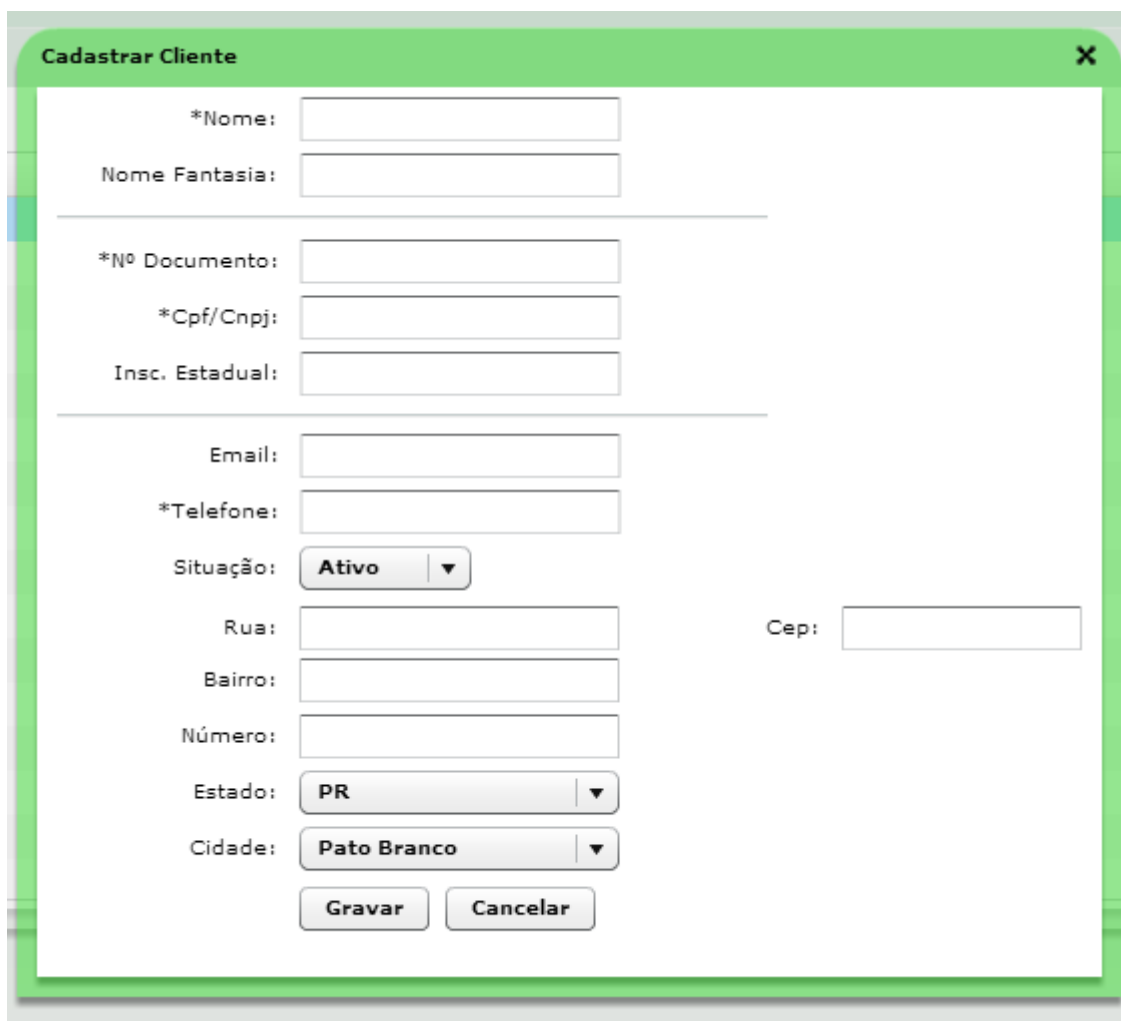


The screenshot shows a window titled 'Clientes' with a close button (X) in the top right corner. Below the title bar are three green buttons: 'Cadastrar', 'Pesquisar', and 'Imprimir'. Below the buttons is a table with the following columns: 'Código', 'Nome', 'Razão Social', 'Email', 'Data Cadastro', 'Situação', 'Telefone', 'Município', and 'Estado'. The table contains one row of data:

Código	Nome	Razão Social	Email	Data Cadastro	Situação	Telefone	Município	Estado
2	Carlos	fantasia	rasador@gmail.com	21/03/2013	Ativo	4691064356	Pato Branco	PR

Figura 15 - Tela de clientes

Ao acionar o botão cadastrar é apresentado um formulário para preenchimento dos dados de um novo cliente, conforme Figura 16. Os campos nome, número de documento, CPF/CNPJ e telefone são de preenchimento obrigatório e o campo CPF/CNPJ possui validação do valor informado, os demais campos não são de preenchimento obrigatório.



O formulário de cadastro de cliente, intitulado "Cadastrar Cliente", possui os seguintes campos e controles:

- *Nome:
- Nome Fantasia:
- *Nº Documento:
- *Cpf/Cnpj:
- Insc. Estadual:
- Email:
- *Telefone:
- Situação:
- Rua:
- Bairro:
- Número:
- Estado:
- Cidade:
- Cep:
- Botões: Gravar, Cancelar

Figura 16 - Formulário de cadastro de cliente

Ao gravar com sucesso na base de dados é exibida uma mensagem de confirmação para o usuário, conforme Figura 17.

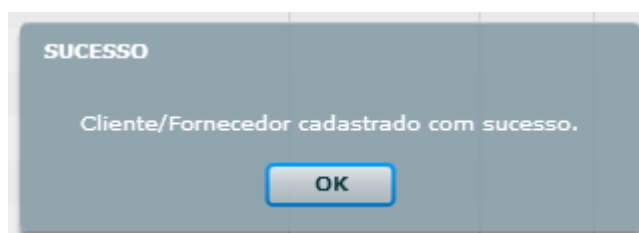
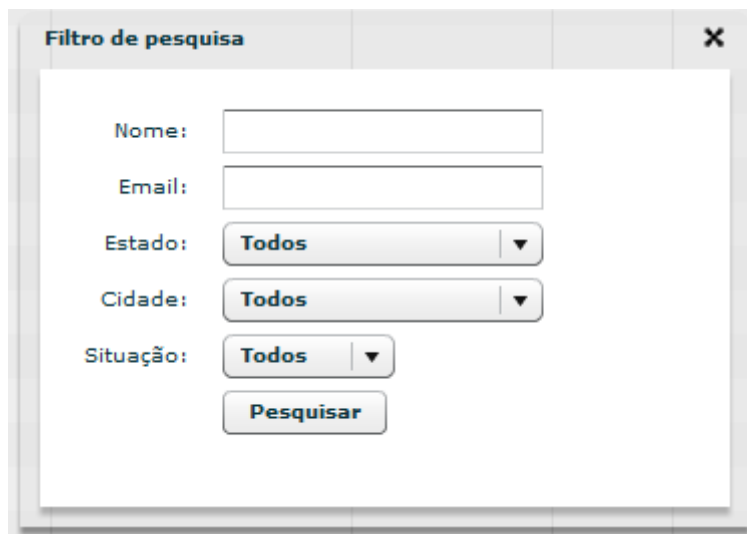


Figura 17 - Mensagem de confirmação de sucesso

Ao acionar o botão pesquisar é apresentada a janela de filtro de pesquisa, conforme a Figura 18, com os campos nome, e-mail, estado, cidade e situação, ao preencher um ou mais campos e acionar o botão pesquisar é realizado uma busca na base de dados.



The image shows a dialog box titled "Filtro de pesquisa" with a close button (X) in the top right corner. It contains the following fields and controls:

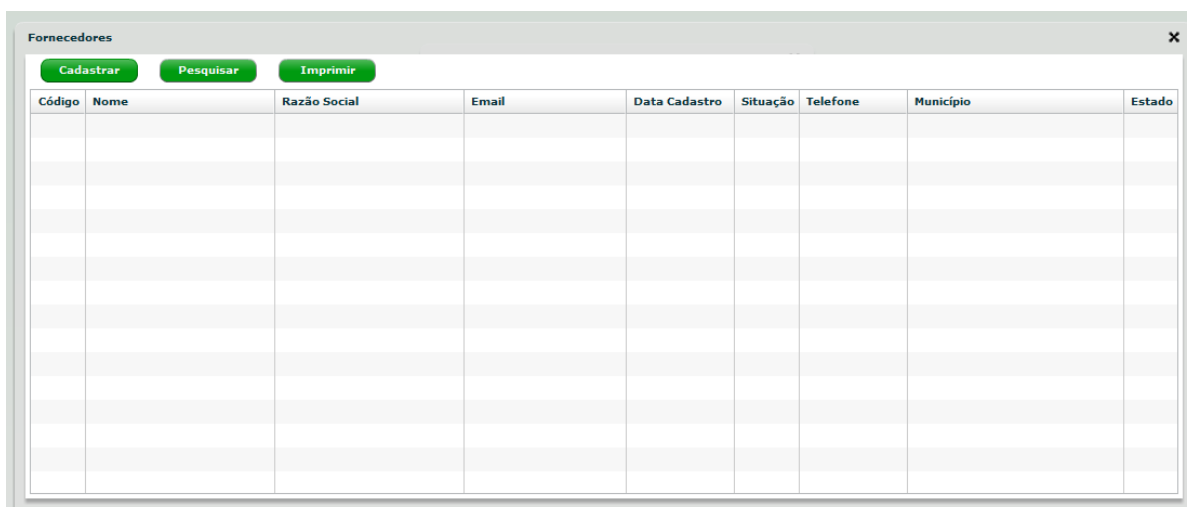
- Nome: [Text input field]
- Email: [Text input field]
- Estado: [Dropdown menu with "Todos" selected]
- Cidade: [Dropdown menu with "Todos" selected]
- Situação: [Dropdown menu with "Todos" selected]
- [Pesquisar button]

Figura 18 - Filtro de pesquisa de clientes

Ao acionar o botão imprimir é apresentado o relatório de clientes contendo os mesmos campos da lista.

4.6.3 Cadastro de Fornecedores

Ao acessar o menu fornecedores é apresentada a tela do cadastro de fornecedores (Figura 19), com a lista de fornecedores cadastrados e com os botões cadastrar, pesquisar e imprimir.



The image shows a window titled "Fornecedores" with a close button (X) in the top right corner. At the top, there are three buttons: "Cadastrar" (green), "Pesquisar" (green), and "Imprimir" (green). Below the buttons is a table with the following columns:

Código	Nome	Razão Social	Email	Data Cadastro	Situação	Telefone	Município	Estado

Figura 19 - Tela de fornecedores

A tela segue o padrão de layout do sistema e contém os mesmos campos e ações do cadastro de clientes, na base de dados os registros são gravados na

mesma tabela, mas com um atributo tipo com valor diferente para identificar cliente e fornecedor. O filtro de pesquisa também tem os mesmos campos que o filtro de clientes e a impressão de relatório possui os mesmos campos apresentados na tela.

4.6.4 Cadastro de Produtos

O cadastro de produtos é exibido ao acionar o menu produtos, a tela apresenta a lista de produtos já existentes na base de dados, com os campos código do produto, descrição do produto, fornecedor, quantidade atual em estoque, quantidade mínima em estoque, valor de compra em reais e valor de venda em reais. No topo da tela são apresentados os botões com as possíveis ações que podem ser realizadas, conforme Figura 20.

Código	Produto	Fornecedor	Quantidade Atu	Quantidade Min.	R\$ Custo	R\$ Venda

Figura 20 - Tela de produtos

Ao acionar o botão cadastrar é apresentado o formulário de cadastro de produto, com os campos produto, código, situação, fornecedor, quantidade mínima em estoque, quantidade atual em estoque, custo de compra e valor de venda,

conforme Figura 21. Os campos produto, quantidade mínima no estoque, quantidade atual, valor de custo e valor de venda são de preenchimento obrigatório.

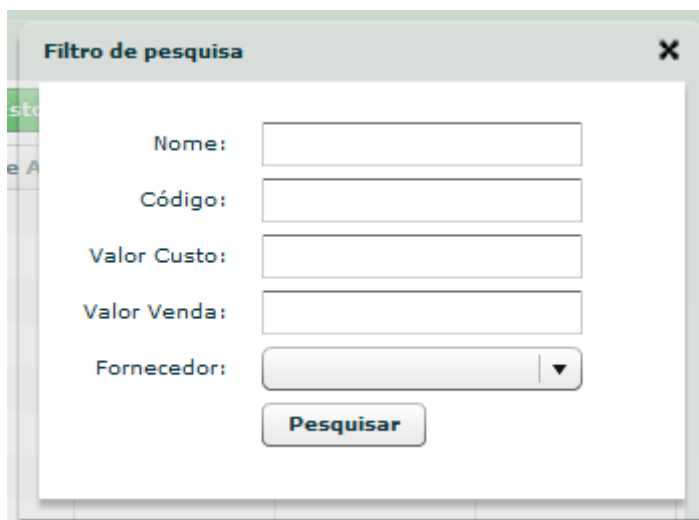


O formulário 'Cadastrar Produto' possui os seguintes campos e controles:

- Produto: Campo de texto.
- Código: Campo de texto.
- Situação: Menu suspenso com o valor 'Ativo' selecionado.
- Fornecedor: Menu suspenso.
- Quantidade Min.: Campo de texto.
- Quantidade Atual: Campo de texto.
- Custo: Campo de texto.
- Venda: Campo de texto.
- Botões: 'Gravar' e 'Cancelar'.

Figura 21 - Formulário de cadastro de produto

Ao acionar o botão pesquisar é apresentada a janela de filtro de pesquisa, conforme a Figura 22, com os campos nome, código, valor de custo, valor de venda, ao preencher um ou mais campos e acionar o botão pesquisar é realizado uma busca na base de dados. Caso nenhum campo tenha sido preenchido e for efetuada uma consulta, retorna todos os registros existentes na base de dados.



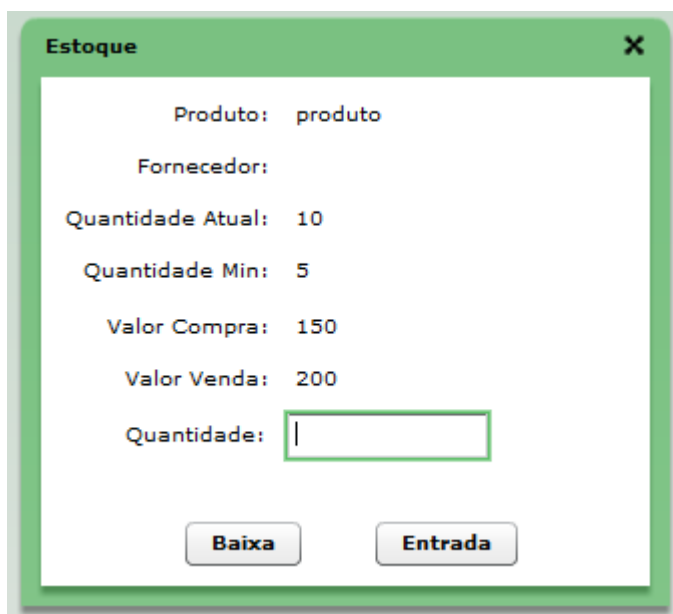
O formulário 'Filtro de pesquisa' possui os seguintes campos e controles:

- Nome: Campo de texto.
- Código: Campo de texto.
- Valor Custo: Campo de texto.
- Valor Venda: Campo de texto.
- Fornecedor: Menu suspenso.
- Botão: 'Pesquisar'.

Figura 22 - Filtro de pesquisa de produtos

Ao acionar o botão imprimir é apresentado o relatório de produtos contendo os mesmos campos da lista. Ao selecionar um registro na lista e acionar o botão

estoque (-) é exibida a tela de estoque, na qual é possível informar uma quantidade e baixar do estoque através do botão baixa ou dar entrada do produto no estoque através do botão entrada, conforme Figura 23.



A imagem mostra uma janela de software intitulada "Estoque" com uma barra de título verde e um ícone de fechar (X) no canto superior direito. O conteúdo da janela é o seguinte:

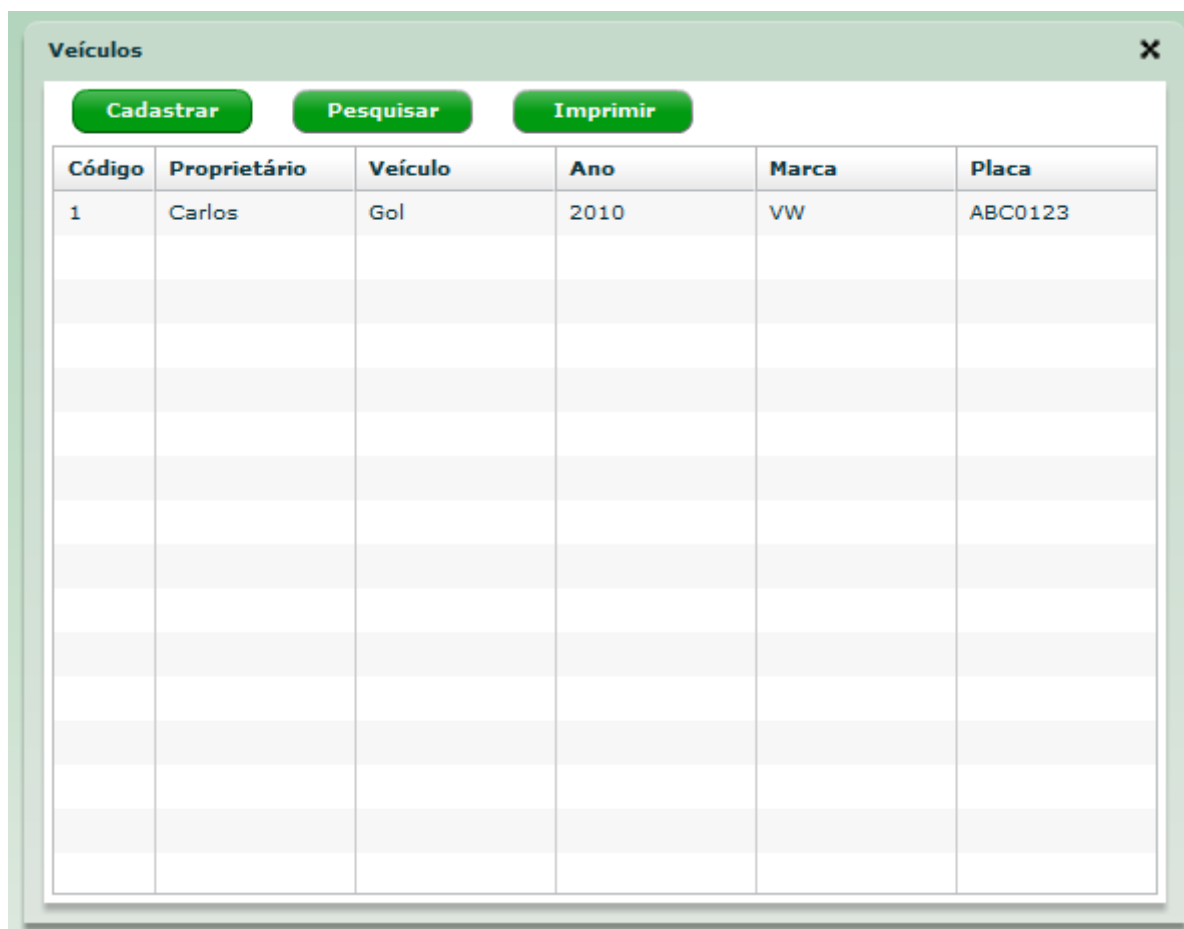
- Produto: produto
- Fornecedor:
- Quantidade Atual: 10
- Quantidade Min: 5
- Valor Compra: 150
- Valor Venda: 200
- Quantidade:

Na base da janela, há dois botões: "Baixa" e "Entrada".

Figura 23 - Tela de estoque

4.6.5 Cadastro de Veículos

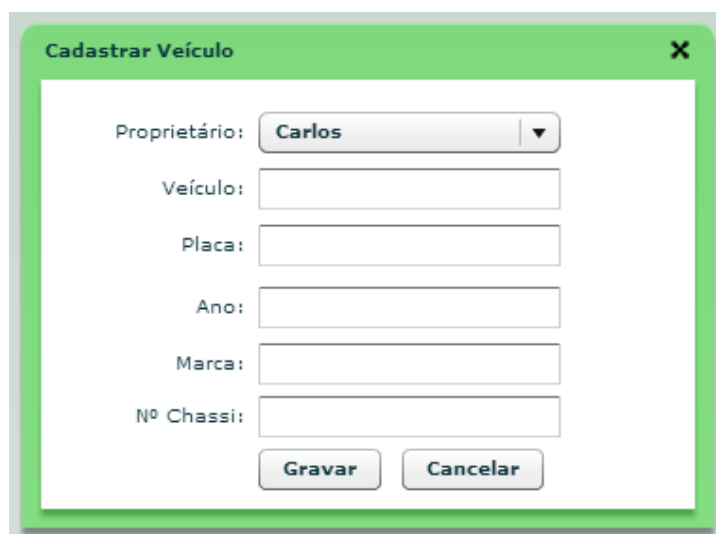
O cadastro de veículos é exibido ao acionar o menu veículos, a tela apresenta a lista de veículos já existentes na base de dados, com os campos código do veículo, nome do cliente proprietário, veículo, ano de fabricação, marca e placa. No topo da tela são apresentados os botões com as possíveis ações que podem ser realizadas, conforme Figura 24.



Código	Proprietário	Veículo	Ano	Marca	Placa
1	Carlos	Gol	2010	VW	ABC0123

Figura 24 - Tela de veículos

Ao acionar o botão cadastrar é apresentado o formulário de cadastro de veículo, com os campos proprietário, veículo, placa, ano, marca e número do chassi, conforme Figura 25. Os campos proprietário, veículo e placa são de preenchimento obrigatório.



Cadastrar Veículo

Proprietário:

Veículo:

Placa:

Ano:

Marca:

Nº Chassi:

Figura 25 - Formulário de cadastro de veículo

Ao acionar o botão pesquisar é apresentada a janela de filtro de pesquisa, conforme a Figura 26, com os campos proprietário, veículo e placa, ao preencher um ou mais campos e acionar o botão pesquisar é realizado uma busca na base de dados. Caso nenhum campo tenha sido preenchido e for efetuada uma consulta, retorna todos os registros existentes na base de dados.

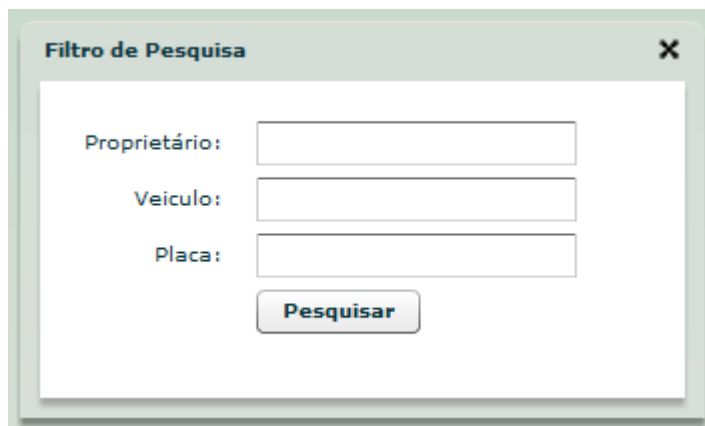
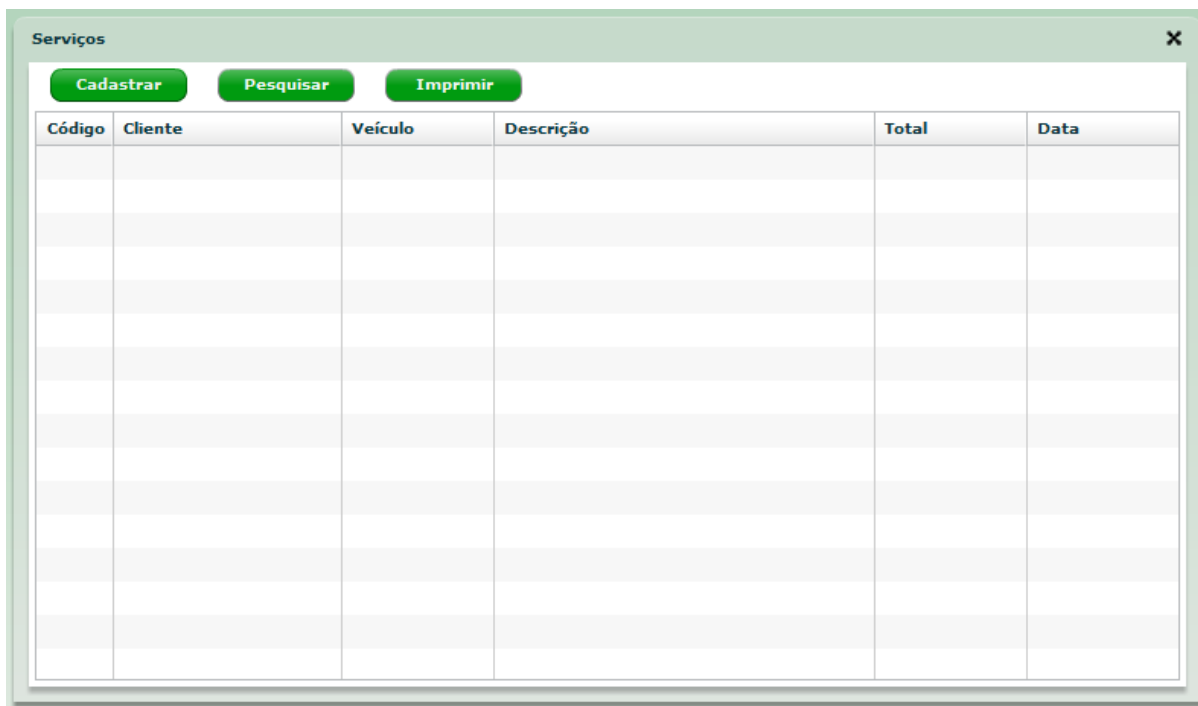
A imagem mostra uma janela de software intitulada "Filtro de Pesquisa" com um ícone de fechar (X) no canto superior direito. O interior da janela contém três campos de entrada de texto, cada um precedido por um rótulo: "Proprietário:", "Veículo:" e "Placa:". Abaixo dos campos, há um botão retangular com o texto "Pesquisar" em azul.

Figura 26 - Filtro de pesquisa de veículos

Ao acionar o botão imprimir é apresentado o relatório de veículos contendo os mesmos campos da lista.

4.6.6 Serviços

O cadastro de serviços é exibido ao acionar o menu serviços, a tela apresenta a lista de serviços já existentes na base de dados, com os campos código, nome do cliente, veículo, descrição, valor total e data. No topo da tela são apresentados os botões com as possíveis ações que podem ser realizadas, conforme Figura 27.



Código	Cliente	Veículo	Descrição	Total	Data

Figura 27 - Tela de serviços

Ao acionar o botão cadastrar é apresentado o formulário de cadastro de serviços, com os combos de seleção de cliente, veículo e produto, com os campos data, quantidade do produto, valor do produto e descrição e com os botões adicionar produto, excluir produto, gravar e cancelar, conforme Figura 28. O formulário também possui a lista de produtos que são incluídos no serviço através do botão adicionar e podem ser removidos selecionando um produto na lista e acionando o botão excluir. Os campos cliente, veículo, descrição e data são de preenchimento obrigatório. Ao acionar um dos botões novo é apresentada a tela de cadastro do item correspondente.

Cadastrar Serviço [X]

Cliente:

Veículo:

Data:

Produto:

Quantidade: Valor:

Produto	Quantidade	Valor Un.	Total

Descrição: Total:

Figura 28 – Formulário de cadastro de serviço

Ao acionar o botão pesquisar é apresentada a janela de filtro de pesquisa, conforme a Figura 29, com os campos cliente e placa, ao preencher um ou mais campos e acionar o botão pesquisar é realizada uma busca na base de dados. Caso nenhum campo tenha sido preenchido e for efetuada uma consulta, retorna todos os registros existentes na base de dados.

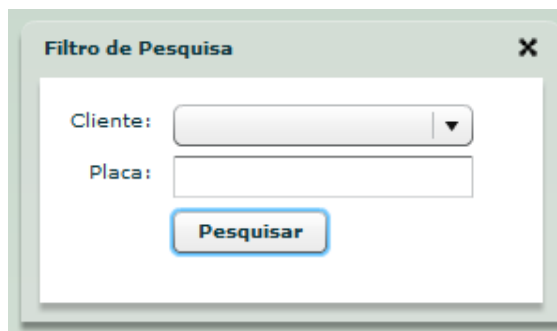


Figura 29 - Filtro de pesquisa de serviços

Ao selecionar um serviço na lista e acionar o botão imprimir é apresentada a ordem de serviço, com os campos do formulário e com o local de assinatura do cliente para autorizar a execução dos serviços.

4.7 IMPLEMENTAÇÃO DO SISTEMA

Nesta seção será descrito como o sistema foi desenvolvido. O sistema foi desenvolvido utilizando a linguagem Java para a implementação no servidor e foi utilizada a tecnologia Flex, com a linguagem ActionScript, para o desenvolvimento das telas e validações do sistema. Para a comunicação entre o Java e o Flex foi utilizado o *framework* BlazeDS e para a persistência de dados foi utilizado o *framework* Hibernate.

Foram criados dois projetos para o desenvolvimento do sistema, um projeto para o servidor Java e outro projeto para o cliente em Flex. O sistema foi construído utilizando a arquitetura MVC. No projeto para o servidor em Java foram criados os pacotes e classes de acordo com a arquitetura MVC, conforme a Figura 30.

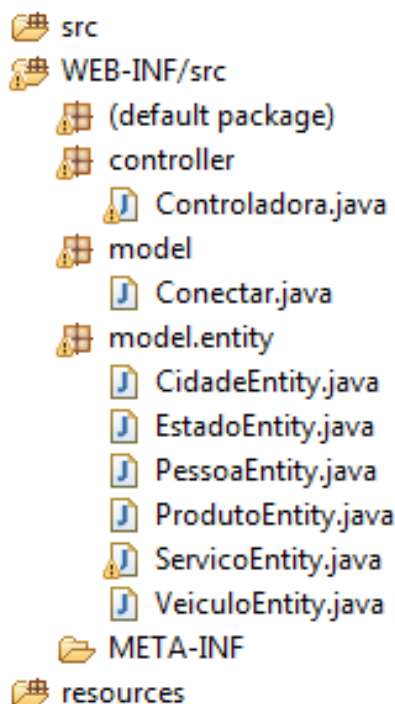


Figura 30 - Pacotes e classes do projeto em Java

Foi necessário importar no projeto as bibliotecas do Hibernate e do MySQL e também configurar um arquivo de persistência no formato xml para o funcionamento do Hibernate, conforme a Figura 31.

```
<persistence-unit name="mysql">
  <provider>org.hibernate.ejb.HibernatePersistence</provider>
  <class>model.entity.PessoaEntity</class>
  <class>model.entity.EstadoEntity</class>
  <class>model.entity.CidadeEntity</class>
  <class>model.entity.ProdutoEntity</class>
  <class>model.entity.VeiculoEntity</class>
  <class>model.entity.ServicoEntity</class>
  <properties>
    <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect"/>
    <property name="hibernate.hbm2ddl.auto" value="update"/>
    <property name="hibernate.show_sql" value="true"/>
    <property name="hibernate.format_sql" value="true"/>
    <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
    <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/mysql"/>
    <property name="javax.persistence.jdbc.user" value="root"/>
    <property name="javax.persistence.jdbc.password" value="root"/>
  </properties>
</persistence-unit>
```

Figura 31 - Arquivo de configuração de persistência de dados

Para a persistência de dados é necessário o mapeamento das classes com suas tabelas correspondentes no banco de dados, cada classe deve ser inserida no arquivo de configuração. Também foram acrescentadas no arquivo algumas configurações padrões do Hibernate e as configurações do *drive* de comunicação, caminho, usuário e senha do banco de dados.

Foi criada uma classe chamada Conectar com dois métodos que auxiliam o controle de transações na aplicação, conforme Figura 32.

```
public class Conectar {

    public static EntityManager entityManager;

    public synchronized static EntityManager getEntityManager(){
        if(entityManager == null){
            EntityManagerFactory factory = Persistence.createEntityManagerFactory("mysql");
            entityManager = factory.createEntityManager();
        }

        return entityManager;
    }

    public static void beginTrasaction(){
        if(!getEntityManager().getTransaction().isActive())
            getEntityManager().getTransaction().begin();
    }
}
```

Figura 32 - Classe conectar

A classe Controladora possui os métodos que controlam as ações na base de dados, como inclusão e alteração de objetos. A Figura 33 mostra um exemplo de como são utilizados os métodos da classe conectar na classe Controladora, no final do método incluiCliFor, que tem a função de incluir um cliente ou fornecedor na base de dados, existe o tratamento para iniciar uma transação, persistir o objeto PessoaEntity e efetuar a transação no banco de dados. Em seguida é feito o tratamento de exceção, caso ocorra algum erro a transação é desfeita e retorna o código de erro, senão é retornado o código de sucesso.

```
try {
    Conectar.beginTrasaction();
    Conectar.getEntityManager().persist(cliente);
    Conectar.getEntityManager().getTransaction().commit();
} catch (Exception e) {
    Conectar.getEntityManager().getTransaction().rollback();
    System.err.println(e);
    return 0;
}

return 1;
```

Figura 33 - Utilização dos métodos da classe conectar

Também é necessário configurar o arquivo remotng-config.xml para o correto funcionamento do BlazeDS, devem ser listados as classes que o Flex irá acessar conforme Figura 34, que mostra a configuração para acesso a classe Controladora e o nome do objeto remoto que o Flex deve acionar, nesse caso é o oficinaService.

```

<service id="remoting-service"
  class="flex.messaging.services.RemotingService">
  <adapters>
    <adapter-definition id="java-object"
      class="flex.messaging.services.remoting.adapters.JavaAdapter"
      default="true"/>
  </adapters>
  <default-channels>
    <channel ref="my-amf"/>
  </default-channels>
  <destination id="oficinaService" >
    <properties>
      <source>controller.Controladora</source>
      <scope>application</scope>
    </properties>
  </destination>
</service>

```

Figura 34 - Arquivo remoting-config.xml

As telas do sistema foram desenvolvidas utilizando Flex e codificadas em *ActionScript*, para cada cadastro foi criada uma classe controladora com os métodos para montar a lista e a pesquisa de registro, validar e salvar o formulário de cadastro. Também foram utilizados diversos componentes do Flex como o *DataGrid* para as listas, o contêiner *TitleWindow* para organizar os campos no formulário de cadastro e o *FormItem* para cada campo do formulário.

Foi criada uma classe chamada *Conexao* para gerenciar as chamadas ao objeto remoto mapeado no arquivo de configuração *remoting-config.xml*, o atributo *destination* do componente *RemoteObject* deve ter o valor do serviço configurado no arquivo. Nesse caso foi *oficinaService* e com o método *efetuaProcesso* passando como parâmetros uma função para tratar erro, uma função para tratar sucesso na operação, o nome do método que será acionado no Java e caso seja necessário os dados parametrizados do método, conforme Figura 35.

```

private static var objetoRemoto:RemoteObject;

public function Conexao():void{
    objetoRemoto = new RemoteObject();
    objetoRemoto.destination = "oficinaService";
}

public function efetuaProcesso(funcaoRetornoErro:Function, funcaoRetornoSucesso:Function,
                               metodo:String, dados:Array):void{

    objetoRemoto.getOperation(metodo).addEventListener(ResultEvent.RESULT, funcaoRetornoSucesso);
    objetoRemoto.getOperation(metodo).addEventListener(FaultEvent.FAULT, funcaoRetornoErro);

    if(dados==null){
        objetoRemoto.getOperation(metodo).send();
    }else{
        objetoRemoto.getOperation(metodo).send(dados);
    }
}
}

```

Figura 35 - Classe conexao

4.8 TESTES DO SISTEMA

Esta seção descreve como foram realizados os testes no sistema. Foram realizados testes informais pelo próprio programador, à medida que o sistema era implementado.

Primeiramente foram codificadas e testadas as telas do sistema em Flex, foi verificado se as telas estavam sendo exibidas corretamente e se todas as funções estavam sendo acionadas de forma correta, como por exemplo, ao acionar o botão gravar no formulário de cadastro de clientes, enviando os dados do formulário como parâmetros para o servidor. Também foram testadas as validações de campos obrigatórios e campos de formato específico, como por exemplo, a validação do número de CPF/CNPJ no formulário de cadastro de clientes.

Por fim foi implementado e testado o servidor em Java, em que foi verificado se os métodos recebiam os parâmetros passados do Flex, tratavam corretamente os dados, efetuavam as operações necessárias na base de dados e retornavam o código de sucesso ou o erro para o Flex.

Com os defeitos descobertos foram realizadas correções e melhorias no código a fim de eliminar erros e falhas encontradas.

4.9 IMPLANTAÇÃO DO SISTEMA

Esta seção descreve como o sistema foi implantado. O sistema foi implantado em um computador com uma configuração mais robusta o qual foi destinado como servidor na empresa do cliente. Neste computador foi instalado o banco de dados MySQL (ORACLE, 2013d) e foi criada a base de dados e também as tabelas apresentada no DER da Figura 11. A base de dados foi configurada com o nome *mysql*, com a senha e *login root*, os mesmos dados utilizados no arquivo de configuração conforme Figura 31. Também foi instalado o servidor Apache Tomcat (APACHE, 2013d). Na pasta de aplicações do servidor chamada *webapps*, foi criada uma pasta chamada *oficina*, onde foi instalado o sistema. Também foi verificado se o Flash Player estava instalado nos navegadores (*browsers*) dos computadores da empresa para executar o sistema. Nos computadores que não estavam instalados foi realizada a instalação.

Por fim os funcionários foram instruídos em como acessar o sistema digitando o endereço IP do servidor no navegador, conforme Figura 36. Para o acesso ao sistema ser feito através de um endereço por nome em vez do endereço IP do servidor, teria que ser instalado e configurado um servidor de DNS (*Domain Name System*), a instalação e configuração está fora do escopo deste trabalho.

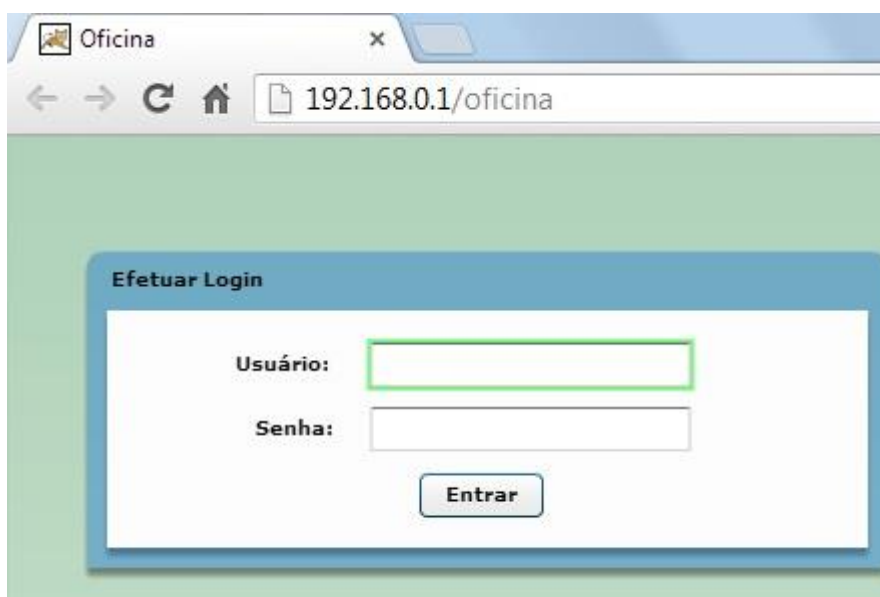


Figura 36 - Endereço IP do sistema

Os funcionários também foram instruídos a utilizar de forma correta o sistema para automatizar o trabalho que era feito de forma manual.

5 CONCLUSÃO

Este trabalho teve como objetivo o desenvolvimento de uma Aplicação Rica para Internet para o gerenciamento de oficinas mecânicas, com a utilização da arquitetura de desenvolvimento MVC.

Os casos de usos necessários para o desenvolvimento deste sistema foram elaborados a partir do levantamento de requisitos, em que foi analisado o funcionamento de uma oficina mecânica para identificar quais as funcionalidades necessárias ao sistema.

Para facilitar a compreensão do sistema foram desenvolvidos também o diagrama de classes e o diagrama de entidade relacionamento, possibilitando assim uma visão geral do sistema.

Após ter a documentação pronta e definidas as tecnologias a serem utilizadas o sistema foi desenvolvido utilizando de técnicas e conceitos como orientação a objetos, modelo cliente/servidor e arquitetura MVC, entre outras. As interfaces do sistema foram elaboradas com tecnologias RIAs, que tornam a experiência de usabilidade do sistema equivalente a proporcionada por aplicativos *desktop*.

O sistema foi devidamente testado, para garantir o correto funcionamento do que foi codificado, foram corrigidos os defeitos e realizadas melhorias na implementação.

Por fim o sistema foi implantado no cliente, em que o servidor foi instalado em uma máquina mais robusta, os funcionários foram instruídos e acompanhados na utilização do sistema.

A utilização deste sistema permitiu o cliente informatizar e melhorar os processos dentro da sua oficina mecânica de forma eficiente e prática, ganhando tempo no atendimento aos clientes e controlando os serviços prestados de forma segura.

Conforme apresentado pode se constatar que o objetivo deste trabalho foi alcançado com sucesso, o qual era realizar a análise e desenvolvimento de uma Aplicação Rica para Internet para gerenciamento de oficinas mecânicas.

REFERÊNCIAS BIBLIOGRÁFICAS

ADOBE. **Adobe BlazeDS**. Disponível em: <<http://opensource.adobe.com/wiki/display/blazeds/BlazeDS>>. Acesso em: 1 mar. 2013a.

ADOBE. **Flash Player**. Disponível em: <<http://www.adobe.com/br/products/flashplayer.html>>. Acesso em: 17 jan. 2013b.

ADOBE. **Adobe**. Disponível em: <<http://www.adobe.com>>. Acesso em: 17 jan. 2013c.

ANICETO, Jefferson. **Aplicações Web**. Apostila ASP.net. Escola Técnica da Univale (ETEIT). 2009.

APACHE. **Apache Flex**. Disponível em: <<http://flex.apache.org/>>. Acesso em: 17 jan. 2013a.

APACHE. **Apache Tomcat**. Disponível em: <<http://tomcat.apache.org/>>. Acesso em: 17 mai. 2013b.

APACHE. **Apache License, Version 2.0**. Disponível em: <<http://www.apache.org/licenses/>>. Acesso em: 17 jan. 2013c.

APACHE. **Compiling and Installing**. Disponível em: <<http://httpd.apache.org/docs/current/install.html>>. Acesso em: 12 mar. 2013d.

BEZERRA, EDUARDO. **Princípios de análise e projeto de sistema com UML**. 2. ed. Rio de Janeiro: Elsevier, 2007.

BOZZON et al. **Conceptual modeling and code generation for Rich Internet Applications**. International Conference on Web Engineering (*ICWE'06*), 2006.

DUHL J. **Rich Internet Applications**. IDC white papers, 2003.

DEITEL, Paul J; DEITEL, Harvey M. **Ajax, Rich Internet Applications e desenvolvimento Web para programadores**. São Paulo: Editora Pearson Prentice Hall, 2008.

ECLIPSE. **Eclipse IDE**. Disponível em: <<http://www.eclipse.org/>>. Acesso em: 17 jan. 2013a.

ECLIPSE. **BIRT Business Intelligence and Report Tools**. Disponível em: <<http://www.eclipse.org/birt/phoenix/>>. Acesso em: 17 jan. 2013b.

FIEL, Jose C. **Entendendo o Flash Remoting**. Disponível em: <<http://rederia.net/2010/08/05/entendendo-o-flash-remoting>>. Acesso em: 12 jan. 2013.

GAMMA, Erich; JOHNSON, Ralph; VLISSIDES, John; HELM, Richard. **Padrões de projetos**. Porto Alegre: Editora Bookman, 2006.

GARRET, Jesse James. **Ajax: A new approach to web applications**. Disponível em: <<http://www.adaptivepath.com/ideas/e000385>>. Acesso em: 29 jan. 2013.

GOOGLE. **Google Web Toolkit**. Disponível em: <<https://developers.google.com/web-toolkit/?hl=pt-BR>>. Acesso em: 17 abr. 2013.

GNU. **GNU Lesser General Public License Version 3**. Disponível em: <<http://www.gnu.org/licenses/>>. Acesso em: 17 jan. 2013.

HUBBARD, John. **Programação com Java**. 2 edição. Porto Alegre: Editora Bookman, 2004.

LAUDON, Kenneth C; LAUDON, Jane P. **Sistemas de informação e a Internet**. Rio de Janeiro: LTC, 1998

MACORATTI, José Carlos. **Padrões de projeto: O modelo MVC**. Disponível em: <http://www.macoratti.net/vbn_mvc.htm>. Acesso em: 19 jan. 2013.

MELO, Ana Cristina. **Desenvolvendo aplicações com UML 2.0: do conceitual a implementação**. 2. ed. Rio de Janeiro: Brasport, 2004.

ORACLE. **JavaFX**. Disponível em: <<http://www.oracle.com/us/products/tools/050854.html>>. Acesso em: 22 jan. 2013a.

ORACLE. **O que é Java?**. Disponível em: <http://www.java.com/pt_BR/download/whatis_java.jsp>. Acesso em: 10 fev. 2013b.

ORACLE. **Why MySQL?**. Disponível em: <<http://www.mysql.com/why-mysql>>. Acesso em: 2 fev. 2013c.

ORACLE. **Installing and Upgrading MySQL.** Disponível em: <<http://dev.mysql.com/doc/refman/5.1/en/installing.html>>. Acesso em: 12 mar. 2013d.

ORACLE. **JavaServer Faces.** Disponível em: <<http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html>>. Acesso em: 3 fev. 2013d.

PRECIADO, J.C. et al. **Necessity of methodologies to model rich internet applications.** Seventh IEEE International Symposium on Web Site Evolution (WSE'05), p. 7-13, 2005.

RED HAT. **Hibernate.** Disponível em: <<http://www.hibernate.org>>. Acesso em: 2 fev. 2013.

RFC2616. **Hypertext Transfer Protocol.** Disponível em: <<http://www.ietf.org/rfc/rfc2616.txt>>. Acesso em: 17 abr. 2013.

SCHMITZ, Daniel P. **Dominando Adobe Flex 4.** Bauru: Editora Viena, 2010.

SERSON, Roberto R. **Programação orientada a objetos com Java 6.** Rio de Janeiro: Editora Brasport, 2007.

WARD, James. **BlazeDS.** Disponível em: <<http://www.jamesward.com>>. Acesso em: 17 jan. 2013.