

FACULDADE SUL-AMERICANA  
SISTEMAS DE INFORMAÇÃO

Ailton Luiz Lima dos Santos  
Ruben Silva Delmondes

**SISTEMA WEB PARA CONTROLE E GERENCIAMENTO DE SERVIÇOS DE  
LAVAGEM DE VEÍCULOS**

GOIÂNIA  
2018

FACULDADE SUL-AMERICANA  
SISTEMAS DE INFORMAÇÃO

Ailton Luiz Lima dos Santos

Ruben Silva Delmondes

**SISTEMA WEB PARA CONTROLE E GERENCIAMENTO DE SERVIÇOS DE  
LAVAGEM DE VEÍCULOS**

Trabalho de Conclusão de Curso  
apresentado à Coordenação do Curso de  
Sistemas de Informação, da Faculdade  
Sul-Americana – FASAM, para  
obtenção do diploma de Bacharel em  
Sistemas de Informação, sob a  
orientação do Prof.º Esp. Telmo  
Queiroz da Silva.

GOIÂNIA

2018

## **FOLHA DE APROVAÇÃO**

Ailton Luiz Lima dos Santos

Ruben Silva Delmondes

SISTEMA WEB PARA CONTROLE E GERENCIAMENTO DE SERVIÇOS DE  
LAVAGEM DE VEÍCULOS

ORIENTADOR:

---

Prof.º Esp. TELMO QUEIROZ DA SILVA

BANCA AVALIADORA

---

Prof.º 1

---

Prof.º 2

---

Prof.º 3

GOIÂNIA

2018

## **DEDICATÓRIA**

Dedicamos este trabalho às nossas famílias que sempre estiveram ao nosso lado e nos deram forças para a completude de mais esta etapa do nosso currículo profissional, acadêmico e de vida.

## **AGRADECIMENTOS**

Agradecemos a todos que nos apoiaram e nos ajudaram de alguma maneira para a evolução acadêmica e feitura deste trabalho.

Ao qualificado corpo docente da faculdade que tanto nos ensinou ao longo do curso, possibilitando a prática do aprendizado neste trabalho.

Ao professor e orientador Telmo Queiroz da Silva, pela confiança, paciência e pelo rico conhecimento transmitido.

Aos nossos colegas e ao meu companheiro de trabalho de conclusão que não mediu esforços para o sucesso deste trabalho.

Aos nossos pais, pelo amor, afeto, carinho e incentivo.

Às nossas esposas por nos apoiar e entender as noites de estudo.

Ao Murilo, dono do lava a jato de veículos, que inspirou esse projeto.

“O que sabemos é uma gota; o que ignoramos é um oceano.”

Isaac Newton

## **RESUMO**

Escreva aqui o resumo do trabalho.

Palavras-chave:

## **ABSTRACT**

Write here your abstract.

Key-words:



## **LISTA DE FIGURAS**

- Figura 1 - Camadas de Engenharia de Software
- Figura 2 - Tela do Software para Lava Car
- Figura 3 - Tela inicial do Gerenciado de Lava-Jato da VE Software
- Figura 4 - Tela do sistema Kad Lavacar
- Figura 5 - Sistema VHSYS
- Figura 6 - Arquitetura cliente servidor conectado à internet
- Figura 7 - Arquitetura cliente servidor em rede interna (sem conexão com a internet)
- Figura 8 - Modelo de arquitetura em camadas para aplicações cliente-servidor
- Figura 9 - Modelo Cascata
- Figura 10 - Modelo em V
- Figura 11 - Modelo Incremental
- Figura 12 - Modelo Espiral
- Figura 13 - Herança Simples e Herança Múltipla
- Figura 14 - Tela de autenticação do sistema
- Figura 15 – Tela dashboard
- Figura 16 - Tela Cadastro de Veículos
- Figura 17 - Tela de Inserção de Novo Veículo
- Figura 18 - Tela de Edição de Veículo
- Figura 19 - Pop-Up de Exclusão de Veículo
- Figura 20 - Tela de Relatórios para o usuário Administrador
- Figura 21 - Tela de dashboard para o usuário perfil Funcionário
- Figura 22 - Tela de Cadastro de Vínculo de Serviço e Funcionário
- Figura 23 - Diagrama de Classes do Sistema Web Proposto
- Figura 24 - Diagrama de Atividade
- Figura 25 - Padrão Model-View-Controller (MVC)
- Figura 26 - Bibliotecas de componentes de JSF
- Figura 27 - Spring Web MVC
- Figura 28 - Gerenciamento do Projeto com o Trello

## **LISTA DE TABELAS**

Quadro 1 - Análise comparativa entre os sistemas

Quadro 2 – Requisitos Funcionais

**SUMÁRIO**

## 1. INTRODUÇÃO

É notável o crescimento da frota brasileira de veículos nas últimas duas décadas. Conforme o relatório do ano de 2013 do Observatório das Metrópoles (2013), uma rede nacional de pesquisa conduzida pelo Ministério da Ciência e Tecnologia e Inovação (MCT&I) e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), a região centro-oeste brasileira conta com quase um veículo para cada três pessoas.

Além do desejo de zelar pelo seu patrimônio particular, a praticidade, o aumento de restrições no uso da água em condomínios e a falta de tempo das pessoas, são fatores que levam os proprietários de veículos automotores a procurarem serviços especializados na lavagem e limpeza destes.

Diante desta demanda crescente e da popularização e evolução da internet e dispositivos eletrônicos, proprietários de estabelecimentos de lavagem de veículos necessitam de ferramentas para gestão dos seus ativos. Mais do que isso, a implantação de um sistema agrega valor a um negócio, torna processos mais eficientes e pode elevar o negócio em questão de mercado.

### 1.1 JUSTIFICATIVA

Com o aumento da frota de veículos e a popularização da internet e de dispositivos computacionais, um estabelecimento de lavagem de veículos, comumente chamado de lava a jato, necessita de um sistema de informação que gerencie seus serviços, clientes, recursos humanos e bens materiais e de consumo.

A proposta deste trabalho é fornecer um sistema para cadastro dos serviços de um lava a jato, bem como as etapas desses serviços, associando cada serviço aos respectivos funcionários executantes, possibilitando cálculos finais de comissão e extratos de relatórios. Incluídos nesse projeto estarão também o cadastro de clientes, funcionários, veículos e serviços.

### 1.2 OBJETIVO GERAL

Desenvolver um sistema de gerenciamento das atividades de um lava a jato, que permita associar cada etapa desses serviços a determinado funcionário, bem como

acompanhar gastos com bens de consumo, fornecendo insumos que permitam o cálculo de comissão e a melhoria contínua e economicidade nos processos.

### 1.3 OBJETIVOS ESPECÍFICOS

- Permitir cadastro de clientes, veículos e funcionários;
- Permitir cadastro das etapas de serviços, associando estas aos funcionários executantes para cálculos de comissões;
- Permitir gerar relatórios de serviços, de comissão e controle de bens materiais;
- Modelar e produzir o software.

O trabalho está organizado da seguinte forma: (resumo dos capítulos\*\*\*).

## **2. FUNDAMENTAÇÃO TEÓRICA**

### **2.1. LIMPEZA DE VEÍCULOS**

As ações de um lava a jato dependem de uma boa gestão, assiduidade e bom desempenho da equipe de trabalho. Uma equipe de um lava a jato movimentado, que permanece em funcionamento durante seis dias da semana, oito horas diárias, necessita de uma equipe de no mínimo seis a oito empregados (CHAMOUN).

A capacitação desses empregados envolve técnicas corretas de limpeza, manuseio de substâncias químicas que podem ser tóxicas para o ser humano e poluentes ao meio ambiente, além de cuidados relacionados à consciência ambiental, principalmente em relação ao consumo de água e energia e o descarte racional de resíduos gerados na atividade (CHAMOUN).

Recepção e entrega do veículo, recebimento pelo serviço, lavagem, secagem, limpeza interna e de tapetes, acabamento, polimento, lubrificação são algumas das atividades em um lava a jato (CHAMOUN). Na recepção e atendimento inicial, uma Ordem de Serviço deve ser gerada, com anotações de possíveis danos já existentes no veículo, observações quanto a objetos em seu interior ou algo que mereça ser registrado, seleção do tipo de serviço que será contratado, inclusão de serviços adicionais, orientação quanto ao tempo de espera entre outras informações que podem ser agregadas. A lavagem externa compreende a limpeza da carroceria com ducha, lubrificação de partes mecânicas, limpeza de pneus, assoalho ou para-lamas. A limpeza interna e acabamento compreenderá aspiração, limpeza de vidros, aplicação de silicone em revestimentos, painéis e tapetes. O veículo, terminado os serviços, será entregue ao proprietário que realizará o pagamento pelos serviços prestados e a ordem de serviço é encerrada.

A gestão, administração e marketing de um estabelecimento de lavagem de veículos incluirão, entre outros, a sua divulgação em mídias diversas, compras, contas a pagar e receber, administração do pessoal, gestão de caixa e banco.

Com o aumento da clientela, do número de funcionários, das compras de insumos materiais e contas a pagar e receber, o empreendedor irá necessitar de uma ferramenta de auxílio à gestão.

Em contato com um proprietário de lava a jato de veículos, o mesmo afirmou da necessidade de estar registrando as etapas dos serviços do lava a jato, bem como os funcionários executantes destas etapas com o fito de se calcular e extrair a comissão de cada

um ao final de determinado período. Foram citadas também algumas funcionalidades desejadas como a gestão de materiais de consumo e a gestão de contas (financeiro), que seriam interessantes ao administrador, se possível possibilitando a geração de relatórios. Tais relatórios, podem ter cunho comparativo, que possibilitem a otimização dos processos do lava a jato, evitando desperdício, aumentando produtividade e auxiliando nas tomadas de decisão.

Com relação ao principal objetivo deste projeto, o cálculo de comissão dos funcionários, comissão é “acréscimo ao salário de um empregado, em função do rendimento do seu trabalho ou dos lucros da empresa; gratificação” (COMISSÃO, DICIONÁRIO).

Empregados comissionados recebem o equivalente ao valor de bens e serviços vendidos por eles. No nosso caso, pelo cumprimento e colaboração em etapas da lavagem de veículos ou execução de determinado serviço no lava a jato. Normalmente, comissões são calculadas com base nos preços de produtos e serviços vendidos. Nada impede que uma empresa use diferentes bases para o cálculo de comissão, como o lucro líquido ou custos de um produto para a companhia. Uma taxa de comissão é atribuída pela empresa, podendo ser ou não diferentes para determinados produtos. Além disso, a taxa de comissão pode ser alterada quando uma quantidade X de determinado produto seja vendido ou, se serviço, executado. A empresa deve determinar o período em que será calculada a comissão, sendo normalmente quinzenal ou mensal. Esse trabalho deverá levar em consideração as comissões compartilhadas, ou seja, quando mais de um funcionário executa a mesma tarefa no lava a jato. Além disso, deverá permitir que o administrador ou gerente participe da comissão caso realize tarefas relativas aos funcionários comissionados. Se mais de um funcionário estiver envolvido em um mesmo serviço, a comissão será dividida entre eles (BRAGG, 2017).

## 2.2 DESENVOLVIMENTO DE SOFTWARE

### 2.2.1 O Software

O software distribui a informação, sendo um produto muito importante nos dias atuais. Ele transforma os dados, os tornando mais úteis em determinado contexto, aumenta competitividade, conecta a rede mundial de computadores e proporciona meios de se obter informações de todas as formas possíveis.

Descrições de software, segundo Pressman (2011, p. 32), poderia ser a seguinte:

Software consiste em: (1) instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho

desejados; (2) estruturas de dados que possibilitam aos programas manipular informações adequadamente; e (3) informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas.

Software é um sistema lógico, enquanto o hardware é físico. O software é um produto que é desenvolvido, ou seja, passa por todo um processo de construção que determinará e concentrará os custos, podendo-se dizer que ele não é fabricado no sentido clássico (PRESSMAN, 2011, p. 32).

Diferentemente do hardware, o software não se desgasta. Enquanto a “curva da banheira” expressa a vida útil do hardware (curva de defeitos apresentados principalmente no início de sua vida útil e após desgastes e intempéries dos componentes eletrônicos), a curva idealizada para o software apresenta altas taxas de defeitos no início de sua vida útil e se estabiliza com o decorrer do tempo. Mas o software pode se deteriorar, com a inclusão de erros durante alterações. Uma maneira de se evitar essa deterioração é a escolha de um projeto melhor de software (PRESSMAN, 2011, p. 33).

A maioria dos softwares são fabricados por encomenda, sendo construídos de forma personalizada. Uma boa prática em um projeto e implementação de um software, é permitir a reutilização de componentes de software em programas diferentes (PRESSMAN, 2011, p. 34).

Atualmente, sete categorias de software imprimem grandes desafios aos engenheiros de software (PRESSMAN, 2011, p. 34):

**Software de Sistema:** Programas feitos para atender outros programas. Possuem forte interação com o hardware do computador e operações de processamento concorrente. Exemplo seriam gerenciadores de arquivos, compiladores, componentes de sistema operacional e drivers.

**Software de Aplicação:** Programas projetados para sanar uma necessidade específica de negócio, processando dados técnicos e operações comerciais, auxiliando tomadas de decisão. Programas que processam vendas e controlam processos de fabricação são softwares de aplicação.

**Software Científico ou de Engenharia:** São softwares que realizam grandes cálculos numéricos, relacionados à diversas áreas de pesquisa como astronomia, vulcanologia, biologia molecular e dinâmicas orbitais de satélites.



**Software Embutido:** É um sistema que reside em um produto ou sistema, implementando funções para o usuário final e para o próprio sistema. Sistema de um forno micro-ondas ou sistema de freios em veículos são sistemas embutidos.

**Software como Linha de Produto:** São programas projetados para diferentes tipos de clientes, podendo ser para um mercado específico e limitado (um sistema de controle de estoque, por exemplo) ou um mercado de consumo em massa (ferramentas de uso pessoal ou comerciais, como editores de texto, planilhas eletrônicas ou gerenciadores de bancos de dados).

**Aplicações para Web:** são aplicações que são abertas por navegadores, os chamados *browsers*, que podem estar integradas à bancos de dados e aplicações corporativas, bem como fornecer funções computacionais, recursos e conteúdo especializados ao usuário final.

**Software de Inteligência Artificial:** São aplicações que em geral não fazem uso de funções numéricas na resolução de problemas complexos. Estão presentes, dentre outras áreas, na robótica, redes neurais artificiais, reconhecimento de voz e imagem, sistemas especialistas e jogos.

### 2.2.2 Engenharia de software

O software está incorporado cada vez mais em nossas vidas, conseqüentemente há uma quantidade maior de pessoas interessadas nas funções oferecidas por uma aplicação. Antes de se desenvolver uma aplicação, deve-se entender o problema.

A complexidade das aplicações atuais demanda uma maior atenção com a interação entre os elementos do sistema. Portanto, projetar é fundamental na engenharia de software (PRESSMAN, 2011, p. 38).

Software devem apresentar qualidade elevada e facilidade de manutenção. Falhas da aplicação podem gerar desde pequenos inconvenientes à situações catastróficas. Segundo Pressman (2011, p.38): “Indivíduos, negócios e governos dependem, de forma crescente, de software para decisões estratégicas e táticas, assim como para controle e operações cotidianas”.

O software durante sua construção passará pelos processos de engenharia. A base da engenharia de software, a sua pedra fundamental, é o foco na qualidade. A estrutura de engenharia de software abrange um processo, um conjunto de métodos e uma gama de ferramentas.

Figura 1 - Camadas de Engenharia de Software



Fonte: PRESSMAN, 2011, p. 39.

Sobre engenharia de software, Pressman (2011, p. 39) esclarece:

A IEEE [IEE93a] desenvolveu uma definição mais abrangente ao afirmar o seguinte: Engenharia de software: (1) A aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção do software; isto é, a aplicação da engenharia ao software. (2) O estudo de abordagens definido em (1).

O processo definirá uma metodologia que permitirá o desenvolvimento do software de forma racional, o controle do gerenciamento do projeto, estabelecimento de marcos, obediência a prazos, garantia de qualidade, gestão de mudanças, sendo aplicados métodos técnicos que produzirão modelos, documentos, formulários, dados, relatórios, entre outros produtos derivados.

O processo que é um conjunto de tarefas, ações e atividades realizadas na consecução de algum trabalho, no contexto de engenharia de software, não deve ser uma prescrição rígida. De maneira oposta, é uma abordagem adaptável a cada equipe, cujo objetivo é a entrega de um produto de qualidade, dentro do prazo, satisfazendo tanto os patrocinadores quanto os clientes (PRESSMAN, 2011, p. 40).

Os métodos de engenharia de software envolvem uma gama de tarefas que incluem comunicação, análise de requisitos, modelagem de projeto, construção do programa, testes e suporte.

As ferramentas da engenharia de software podem automatizar o processo ou os métodos. Quando as ferramentas são integradas, ou seja, interagem entre elas fornecendo um suporte ao desenvolvimento de software, ocorre a chamada engenharia de software com auxílio de computador (PRESSMAN, 2011, p. 40)..

### 2.2.3 Aplicações baseadas na Web (WEBAPPS)

Por volta dos anos de 1990 a 1995, a internet era baseada apenas em páginas *HyperText Markup Language* (HTML), um conjunto de arquivos de hipertexto com linguagem de marcação, que interligados apresentavam informação limitada. Com o tempo, foi possível que engenheiros da internet mesclassem outros recursos computacionais, nascendo as aplicações baseadas na Web. Pressman (2011, p. 37), assim comenta sobre as aplicações para a Web:

Atualmente, as WebApps evoluíram para sofisticadas ferramentas computacionais que não apenas oferecem funções especializadas (*stand-alone functions*) ao usuário final, como também foram integrados aos bancos de dados corporativos e às aplicações de negócio. (PRESSMAN, 2011, p. 37).

Uma aplicação Web envolve publicação impressa e desenvolvimento de software, arte e tecnologia respectivamente, com comunicações internas e relações externas (PRESSMAN, 2011, p. 37).

Um WebApp fará uso intensivo de redes, possibilitando acesso mundiais ou mais restritos (intranet), além de simultaneidade de conexões. Esse número de conexões é imprevisível e mesmo assim um WebApp deve possuir um bom desempenho e garantir disponibilidade. Uma aplicação Web é orientada a dados, disponibilizando ao usuário, texto, gráficos, áudio, vídeo, além de serem comumente utilizados no acesso à informação de bancos de dados. A qualidade estética desses sistemas é importante, determinando muitas vezes sua qualidade. Também são atributos de sistemas Web a sua evolução contínua, imediatismo (tempo de produção e implantação) e possuir requisitos mais rígidos com relação à segurança, como limites de número de usuários, proteção à conteúdo e modos de transmissão (PRESSMAN, 2011, p. 37 e 38).

## 2.3. ANÁLISE COMPARATIVA

Nesta subseção são analisados brevemente alguns sistemas similares ao proposto neste trabalho, sendo essa uma análise exploratória e comparativa. A cartilha de “Como montar um lava a jato” do Serviço Brasileiro de Apoio às Micro e Pequenas Empresas - SEBRAE (CHAMOUN, 2012), sugere os sistemas “Kad Lavacar – Software para lava

rápido” e “Datacross – Programa para controle de lava rápido”. Os links especificados nessa cartilha estão desatualizados, sendo necessário a procura na internet de sistemas similares. Encontramos as seguintes soluções na internet: “Software para Lava Car” da Agência Solutions Software, “Gerenciador de Lava-Jato da VE Software”, “Kad Lavacar” e o sistema “VHSYS”.

### 2.3.1 Software para Lava Car da Agência Solutions Softwares

O sistema “Software para Lava Car” da empresa Agência Solutions Softwares é um sistema *desktop* para sistema operacional Windows e que possibilita o uso em rede. O “Software para Lava Car” é um sistema pago, sem mensalidades, entregue via *Compact Disk* (CD) por serviço postal e oferece como ferramentas o atendimento de lavagem de veículo, cadastro de clientes, agenda, caixa (financeiro), fornecedores, contas a pagar e receber, mecanismo de *backup* e senhas por módulo.

Figura 2 - Tela do Software para Lava Car

The screenshot displays the 'Lava-Carro' software interface. At the top, there is a menu bar with options: Agenda, Clientes, Fornecedores, Produtos, Atendimento, Caixa, C.Pagar, C.Receber, and Operacional. Below the menu is a toolbar with various icons. On the right side of the toolbar, there is a login section with a 'Senha:' label, a password input field, and a 'Logar' button. The main window is titled 'Clientes - Cadastro' and contains a sub-toolbar with buttons: Grava, Limpa, Altera, Apaga, and Volta. To the right of these buttons are fields for 'Cadastros:' (showing '0'), 'Navegador:' (with navigation arrows), and 'Código:'. The main form area is divided into sections for data entry. The first section has tabs for '1ª Parte', '2ª Parte', and 'Pesquisa'. It includes fields for 'Nome:' (with a dropdown and search icon), 'Contato:', 'Data:' (pre-filled with '31/10/2010'), 'Telefone 1:', 'Telefone 2:', 'Telefone 3:', and 'E-mail:'. The second section contains fields for 'Endereço 1:', 'Bairro:', 'Cidade:', 'Estado:' (with a dropdown), and 'Cep:'. The third section has 'Carro:' (with a dropdown) and 'Placa:'. The bottom section includes 'CNPJ ou CPF:', 'RG. ou Ins. Estadual:', 'Data Nasc.:' (pre-filled with '//'), and 'Grupo:' (with a dropdown). At the bottom of the window, there is a taskbar showing several open application windows.

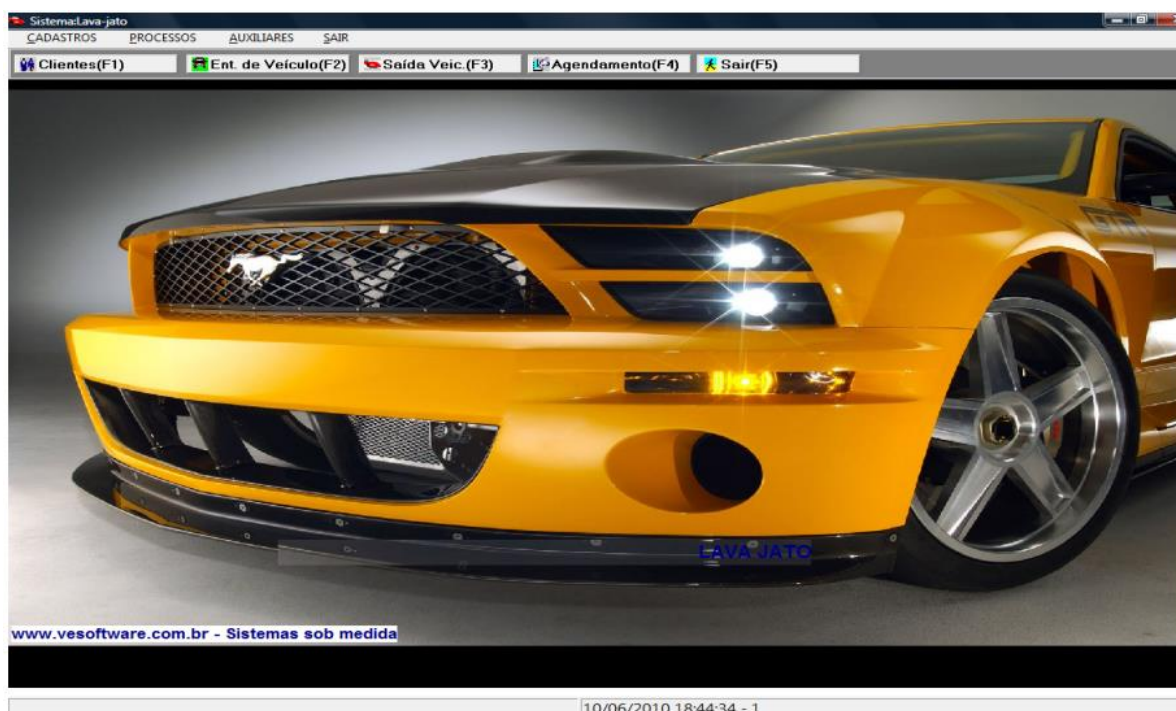
Fonte: Sítio da Aplicação

### 2.3.2 Gerenciador de lava-jato da VE Software

O “Gerenciador de Lava-Jato” da VE Software pode ser instalado em um servidor e usado por computadores em rede. É um sistema *desktop* para o sistema operacional Windows, pago, podendo ser entregue por *e-mail* ou por serviço postal via *Compact Disk* (CD). O sistema possui flexibilidade no uso impressoras e possui como funcionalidades:

- Cadastro de clientes, veículos, usuários;
- Recebimentos mensais dos clientes, com multa por atraso, recibo de pagamentos;
- Convênios com possibilidade de descontos em dinheiro ou porcentagem;
- Conta Corrente de Funcionário;
- Controle fácil de pátio, mostrando quais carros estão estacionados, número do box e chave;
- Backup Automático;
- Controle de lavagens ou serviços;
- Controle de pontos por lavagem, avisando quando pode trocar os pontos de fidelidade.

Figura 3 - Tela inicial do Gerenciado de Lava-Jato da VE Software



Fonte: Sítio da Aplicação

### 2.3.3 Sistema Kad Lavacar

O sistema “Kad Lavacar” da empresa Enkad Tecnologia é um sistema *desktop* para o sistema operativo Windows que pode funcionar em rede (até dez computadores) e fornece ao usuário a opção de alteração de temas na sua *interface*. Em sua versão 5.08, o “Kad Lavacar” é um sistema pago, entregue via e-mail e oferece como ferramentas:

**Cadastros:** transportadoras, fornecedores, funcionários, vendedor, produtos, serviços, clientes.

**Movimentos:** serviço/venda.

**Relatórios:** placas por clientes, serviços a executar, serviços executados, serviços a executar (por cliente), serviços executados (por cliente), contas a receber (do cliente), contas a receber (geral), estoque, vendas ao cliente, clientes aniversariantes, vendas por vendedor, comissão por vendedor, vendas por período.

**Listas:** produtos, clientes, serviços, vendedores, funcionários, fornecedores, transportadoras.

**Consultas:** preços serviços, preços produtos, vendas ao cliente, vendas/serviços, recebimentos.

**Financeiro:** caixa, contas, despesas, cheques, funcionários, agenda hoje.

**Relatórios do Financeiro:** caixa do dia, caixa período, entradas, saídas, contas a pagar hoje, contas a pagar, contas vencidas, contas pagas, despesas por dia, despesas por período, cheques recebidos hoje, cheques recebidos por período, cheques recebidos por status, cheques recebidos vencidos, cheques recebidos a vencer, cheques emitidos hoje, cheques emitidos a vencer hoje, cheques emitidos a vencer, cheques emitidos vencidos, funcionários (pagamento a realizar), funcionários (pagamento realizado), agenda hoje.

**Diversos:** compras, anotações, imprimir anotações, contatos, agenda, promissória avulso, recibo avulso, carta cobrança.

**Help Desk:** cadastro de departamentos, *help desk*, consulta, relatórios.

**Arquivo:** construção de consultas, reparação, configuração impressora, manutenção de senhas, *backup*.

Figura 4 - Tela do sistema Kad Lavacar

KAD LAVACAR Versão: 5.06

Cadastros Movimentos Relatórios Listas Consultas Financeiro Diversos Help Desk Arquivo Editar Exibir Janela Ajuda Backup SKIN

Transportador Fornecedores Funcionários Vendedores Produtos Serviços Clientes Venda/Serv

**Venda / Serviços**

Incluir Gravar Excluir Procurar

Código  Data

Vendedor  Observações

Cliente

SERVIÇOS							
Placa	Data	Serviços	Quantidade	Valor R\$	Desconto R\$	Acréscimo R\$	Total itens
*							

Imprimir Venda / Serviços Imprimir Recibo

Total R\$  Recebido R\$  Débito R\$

Fonte: Sítio do desenvolvedor da aplicação

### 2.3.4 Sistema VHSYS

O sistema VHSYS é um sistema *online* de gestão empresarial pago mensalmente. É um sistema que possui uma maior gama de funcionalidades em comparação com os sistemas apresentados anteriormente. Por ser *online*, ele garante mobilidade, podendo ser acessado a qualquer lugar por *tablets*, *smartphones* e computadores. As principais ferramentas do sistema são:

**Financeiro** (Contas a Pagar, Contas a Receber, Centro de Custo, Fluxo de caixa, Custo Fixo mensal, Conciliação Bancária, Emissão de Recibo, Emissão de Boleto com registro, Emissão de Boleto sem registro, Emissão de Duplicata Mercantil, Envio de Faturas por e-mail, Condições de Pagamento Customizáveis);

**Vendas e faturamento** (Orçamentos, Pedidos, Geração de ordem de compra com base no pedido, Agrupamento de pedidos, Ordem de produção, Nota Fiscal Eletrônica (NF-e),

Nota Fiscal Eletrônica de Importação e Exportação, Guarda de XML durante 6 anos, Envio de e-mail de Orçamento, Pedido e Nota fiscal, Carta de Correção Eletrônica, Inutilização de notas fiscais, Emissão de Etiquetas de produtos, Emissão de Etiquetas de transporte, Importação de XML ou TXT padrão SEFAZ, Emissão e Impressão de Notas Fiscais em massa, Controle da situação da nota fiscal, Lançamento e gerenciamento de comissões, Venda Balcão e impressão de cupom não fiscal);

**Compras** (Ordem de compra, Entrada de mercadoria, Lançamento de estoque, Importação de XML do fornecedor, Cadastramento de produtos, fornecedores, transportadoras via XML, Guarda de XML durante 6 anos, Nota fiscal eletrônica de entrada, Formação de Preço);

**Estoque** (Controle de estoque, Movimentação de estoque diário e mensal, Custo médio de produtos, Controle de Kits, Baixa de matéria prima, Controle de baixa manual ou automático, Posição do estoque, Emissão de etiquetas);

**Serviços** (Ordem de serviço (OS), Controle técnico de horas, Envio de e-mail, Status da execução da Ordem de serviço, Emissão de nota fiscal de serviço e venda com base na OS, Emissão de nota fiscal de serviço (NFs-e), Guarda de XML durante 6 anos, Exportação de XML padrão ABRASF, Controle da situação da nota fiscal, Serviços recorrentes, Geração de Boletos no Serviço recorrente, Geração de Nota Fiscal no Serviço recorrente, Histórico de serviços);

**Relatórios** (Cadastros de clientes, fornecedores, produtos, transportadoras, Movimentação de Estoque, Controle financeiro de entrada e saída, Faturamento diário, semanal, mensal e anual, Comissões sobre títulos ou valor integral, Produtos mais vendidos, Entrada de mercadoria, Controle de serviços, Evolução de vendas mensal, Margem de Lucro simples, Margem de Lucro detalhado, Controle de Frete, Clientes que mais compram);

**Loja de Aplicativos (APPs):** São disponibilizados aplicativos com novos recursos ao cliente. Alguns aplicativos disponíveis, já realizam: Backup de notas fiscais, Central de Avisos, Agenda integrada, Interação com Contabilidade e Emissor de DARF.

**WebServices:** Integrações com o ERP do cliente, cadastro de produtos, clientes e vendas.

**Envio de SMS:** O sistema possui um modulo de envio de sms, para qualquer operadora de telefonia celular no Brasil. Valores de envio de sms são adicionados ao pacote mensal.

**Consulta Serasa:** O sistema possui um modulo de envio de Consulta SERASA. As consultas são cobradas a parte;



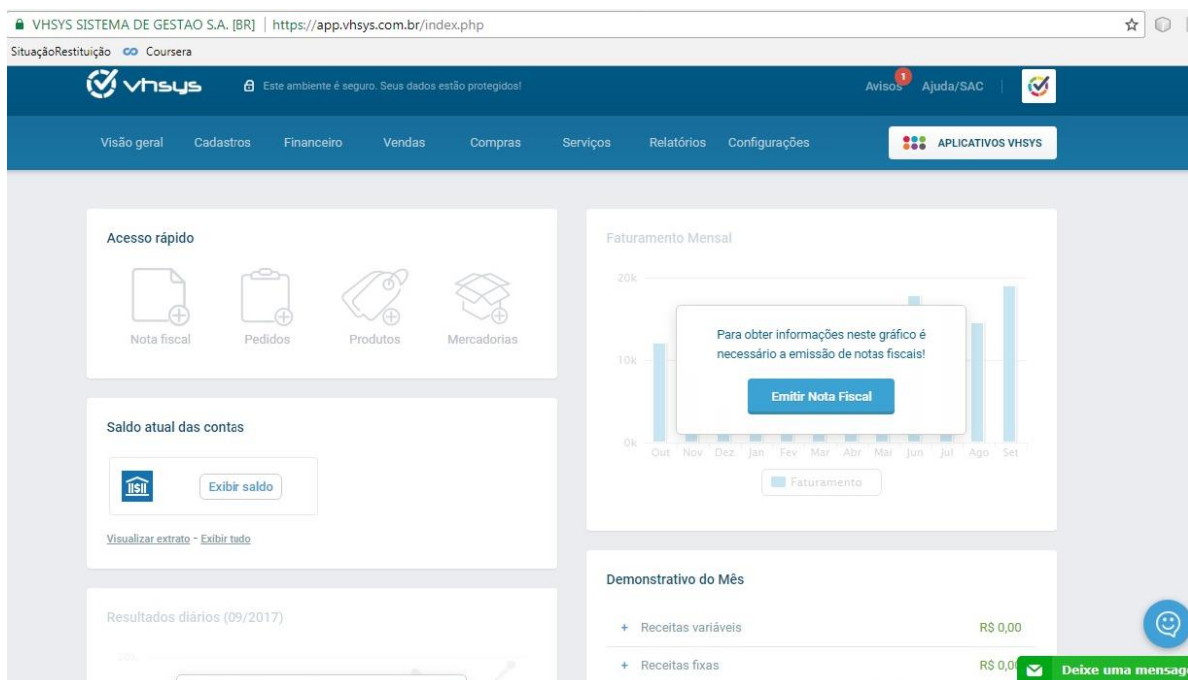
**Loja Virtual:** O sistema possui um modulo que possibilita sua empresa vender pela internet;

**Aplicativo de Pesquisas:** Cria pesquisas e avalia a satisfação dos clientes, podendo criar várias pesquisas simultaneamente, definir datas de apuração, e organizar perguntas.

**Aplicativo de Envio de E-mail Marketing;**

**Aviso de Aniversariantes.**

Figura 5 - Sistema VHSYS



Fonte: *Print Screen* da aplicação.

Quadro 1 - Análise comparativa entre os sistemas

Funcionalidades	Sistema Proposto	Software para Lava Car da Agência Solutions Software	Gerenciador de Lava-Jato da VE Software	Kad Lavacar	VHSYS
Cadastro de Veículos	Sim	Sim	Sim	Sim	Sim
Cadastro de Clientes	Sim	Sim	Sim	Sim	Sim
Controle de Estoque	Sim	Não	Não	Sim	Sim
Controle Financeiro	Não	Sim	Não	Sim	Sim
Pesquisa de Satisfação	Não	Não	Não	Não	Sim
Relatórios Gerenciais	Sim	Não	Não	Sim	Sim
Campanhas de Marketing e Publicidade	Não	Não	Não	Não	Sim
Envio de SMS	Não	Não	Não	Não	Sim
Envio de Email	Sim	Não	Não	Não	Sim
Remuneração e Comissionamento Analítico	Sim	Não	Não	Não	Não
HelpDesk	Não	Não	Não	Sim	Não
Módulo de Backup Automático	Não	Não	Sim	Sim	Não
Controle de lavagens e serviços	Sim	Sim	Sim	Sim	Sim
Agendamento	Não	Sim	Não	Sim	Não
Listagens/Consultas	Sim	Não	Sim	Sim	Sim
NF-e	Não	Não	Não	Não	Sim
Sistema Web	Sim	Não	Não	Não	Sim

Fonte: elaboração dos autores (2018)

Com base no quadro comparativo dos sistemas de lava a jato, fica evidente que o sistema proposto possui funcionalidades importantes que já estão presentes em alguns dos sistemas apresentados.

## 2.4 METODOLOGIA

Métodos são fases ou etapas a serem seguidas com o intuito de se alcançar um objetivo. Essas fases podem ser conceituadas como um conjunto de procedimentos intelectuais e técnicos. O método científico se refere a um:

[...] conjunto de atividades sistemáticas e racionais que, com maior segurança e economia, permite alcançar o objetivo – conhecimentos válidos e verdadeiros – traçando o caminho a ser seguido, detectando erros e auxiliando as decisões do cientista. (MARCONI, LAKATOS, 2010, p. 65).

O método proposto será conduzido de forma **indutiva**, pois a partir do relato e entrevista do proprietário de um estabelecimento de limpeza de veículos, tomou-se iniciativa da motivação do desenvolvimento desse trabalho. O conhecimento é baseado na experiência, partindo de condições particulares às generalizações da realidade.

**Definição do tema:** O tema escolhido surgiu através do contato com o proprietário de um lava a jato de veículos, que possuía a necessidade de controlar os serviços realizados no empreendimento, bem como, registrar a execução dos serviços e suas fases por seus funcionários.

**Caracterização da pesquisa:** Foram definidos os métodos do trabalho, suas delimitações e propostas alternativas de solução da problemática relatada.

**Referências bibliográficas:** Foram definidas referências bibliográficas que dariam apoio na proposta de solução da problemática no trabalho.

**Definições de requisitos e arquitetura do software:** Nesta etapa, modelamos a solução, através de diagramas UML e prototipagem e definimos algumas tecnologias a serem utilizadas pra resolução da problemática.

**Desenvolvimento do software:** Após estabelecimento de referências bibliográficas e definição de requisitos, modelagem e tecnologias a serem utilizadas, passamos a executar a engenharia de software, codificando a solução.

**Conclusão:** Sabendo da necessidade do empreendedor na melhoria do seu negócio, viabilizando-se uma ferramenta que permita mensurar a produção no todo e individual dos seus funcionários e do negócio em si, poderemos com a utilização da solução apresentada, mensurar o benefício ao negócio apresentado.

**Trabalhos futuros:** Serão descritas novas funcionalidades que poderão ser adicionadas à solução proposta no trabalho bem como algumas melhorias.

A pesquisa, conforme Marconi, Lakatos, 2008, p. 1, “[...] é um procedimento formal, com método de pensamento reflexivo, que requer um tratamento científico e se constitui no caminho para se conhecer a realidade ou para descobrir verdades parciais”.

Quanto à natureza, a técnica de pesquisa será **aplicada**, caracterizando-se pelo interesse prático, sendo seus resultados aplicados na solução de problemas específicos.

Quanto aos objetivos, a pesquisa buscará familiaridade com o problema e será **exploratória**, pois envolverá levantamentos bibliográficos, entrevistas e análises de caso.

O trabalho se classificará quanto aos procedimentos técnicos, como **pesquisa bibliográfica** e **pesquisa de campo**, exploração de obras escritas e interação entre os envolvidos, respectivamente. Na busca pela familiaridade com relação ao problema, se aprofundando em questões subjetivas do fenômeno, coletando informações em entrevistas e documentos, tendo um caráter exploratório e não utilizando métodos estatísticos, a pesquisa terá uma abordagem **qualitativa**.

### 2.4.1. Recursos financeiros

Nesta seção apresentaremos os gastos financeiros com o projeto proposto.

Descrição dos Gastos	Quantidade	Preço

### 2.4.2. Cronograma

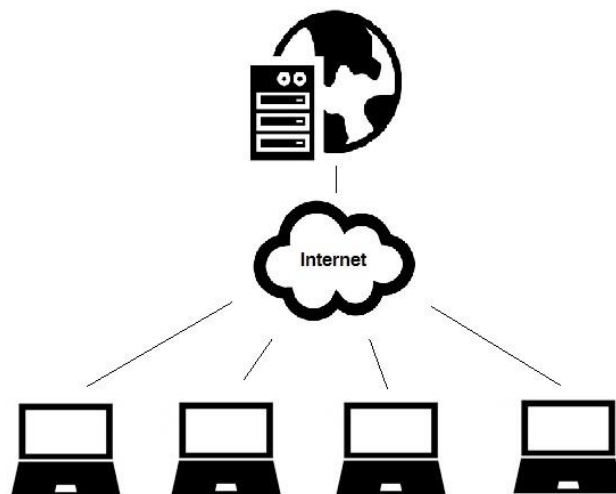
A seguir apresentamos o cronograma do projeto proposto.

Ano	2017					2018					
Descrição das Fases	Ago	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun
Definição do Tema	X										
Procurar Orientador	X										
Caracterização da Pesquisa		X									
Referências Bibliográficas		X	X								
Apresentação TCC1			X								
Definições de Requisitos			X								
Definições de Arquitetura				X							
Desenvolvimento do Software					X	X	X	X			
Conclusão e Trabalhos Futuros									X		
Banca e Qualificação TCC2										X	
Revisão Trabalho Escrito									X	X	
Apresentação TCC2											X

### 2.4.3. Proposta de solução

O sistema será executado em uma arquitetura cliente-servidor, multiusuário e baseado na Web. Seu acesso se dará via navegador (*browser*), a partir de qualquer local caso esteja hospedado em algum servidor na internet. Essa característica é importante ao cliente, proprietário do lava a jato caso haja necessidade de gerenciar mais de um negócio ou empresa à distância.

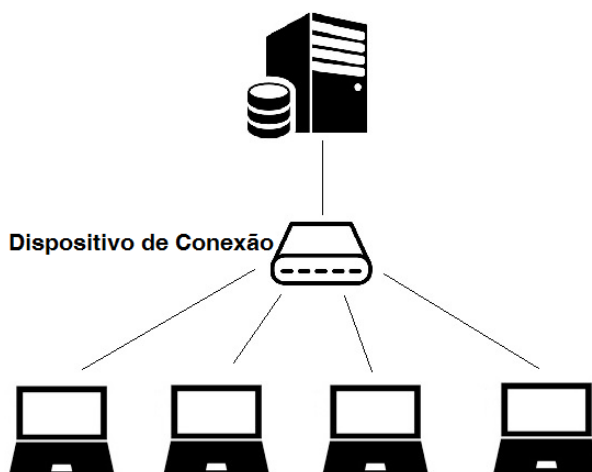
Figura 6 - Arquitetura cliente servidor conectado à internet



Fonte: Elaboração dos autores (2018).

É possível também a utilização do sistema Web sem a conexão com a internet, ocorrendo a conexão apenas em uma rede interna, utilizando-se de algum equipamento de distribuição e conexão de rede, como *hubs*, *switches*, roteadores, seja essa conexão física ou via rádio. O equipamento de conexão e distribuição de rede será o responsável pela conexão dos usuários/clientes ao servidor da aplicação Web.

Figura 7 - Arquitetura cliente servidor em rede interna (sem conexão com a internet)

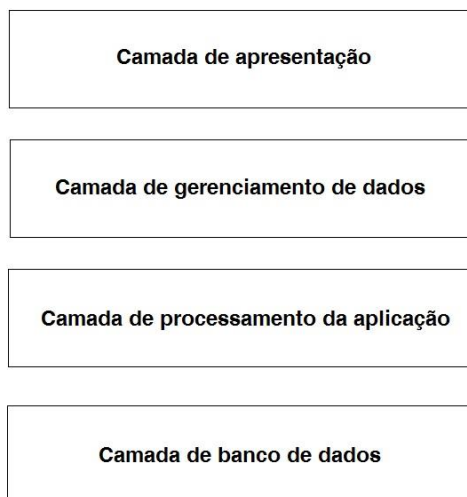


Fonte: Elaboração dos autores (2018).

Segundo Sommerville (2011, p. 340), o modelo cliente-servidor implementa o conceito de software como um serviço (*SaaS*, do inglês *software as a service*). Software como um serviço é um conceito importante, que está em crescimento e implementa uma aplicação

estruturada em quatro camadas, sendo elas a camada de apresentação (apresenta e gerencia as informações ao usuário), camada de gerenciamento de dados (camada que controla e verifica os dados recebidos e passados ao usuário), camada de processamento de aplicação (implementa a lógica e as funcionalidades para o usuário) e camada de banco de dados (armazena e gerencia os dados).

Figura 8 - Modelo de arquitetura em camadas para aplicações cliente-servidor



Fonte: Baseado em Sommerville (2011, p. 340)

#### 2.4.4. Delimitação do trabalho

Algumas delimitações do trabalho se fazem necessárias devido ao tempo limitado, impondo-nos um foco maior em determinadas implementações do projeto. Portanto, algumas delimitações são:

- O sistema pretende atender aos proprietários e funcionários de lava a jatos, bem como promover algumas funcionalidades que dizem respeito aos usuários;
- O desenvolvimento desse trabalho é realizar uma modelagem, protótipo e codificar grande parte do sistema;
- Não é foco do trabalho o desenvolvimento e estudo de mecanismos de segurança da informação;
- O usuário deverá estar conectado ao servidor que hospeda o sistema;
- Esse projeto não considerará questões de salvaguarda de dados e *backup*;
- Não é foco do trabalho o desenvolvimento de um manual referente às funcionalidades do sistema;

- Não é foco do trabalho a presunção de uma estrutura para o correto funcionamento do sistema.

### 3. ANÁLISE DA SOLUÇÃO

#### 3.1 PROCESSOS DE SOFTWARE

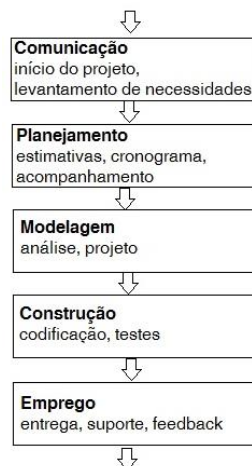
Para Sommerville (2011, p. 18), processos de software não possuem normas rígidas, podendo ser adaptados às características da equipe de desenvolvimento ou do sistema proposto. Mas esses processos, mesmo diferentes e adaptáveis, possuem como ponto em comum algumas atividades como especificação (análise requisitos), projeto, implementação (codificação), validação (testes e garantia que o software realize o que o cliente deseja) e evolução de software (atendimento de novas necessidades do cliente) .

##### 3.1.1 Alguns modelos de processo de software

###### 3.1.1.2 Modelo em cascata

O modelo em cascata, também chamado de ciclo de vida clássico do software, é o paradigma mais antigo da engenharia de software. Ele define uma metodologia sistemática e sequencial (PRESSMAN, 2011, p. 59), iniciando com a comunicação e levantamento das necessidades do cliente, planejamento (estimativa e cronograma), modelagem (análise e projeto), construção (codificação e testes) e emprego (entrega, suporte e *feedback*).

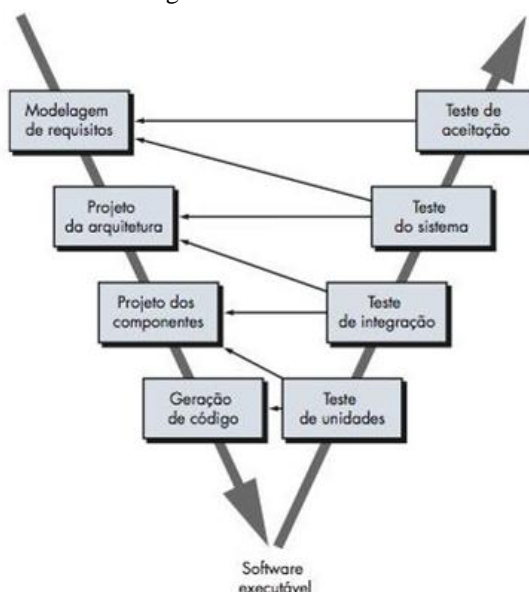
Figura 9 - Modelo Cascata



Fonte: Baseado em Pressman (2011, p. 60).

O modelo em V é resumidamente uma representação do modelo em cascata com indicação de testes a serem aplicados concomitantemente à execução das etapas descritas no modelo em cascata. Esse modelo visa ações de garantia de qualidade ao modelo cascata. Após a execução de cada etapa contida no modelo cascata, são executados os testes às referidas etapas, conforme a figura XX.

Figura 10 - Modelo em V



Fonte: PRESSMAN, 2011, p. 60.

O modelo em cascata se adapta melhor a projetos com requisitos bem definidos, que não mudam radicalmente e possibilitam o desenvolvimento de forma linear (PRESSMAN, 2011, p. 61), se tornando impraticável aos trabalhos com mutações intermináveis (em requisitos, regras de negócios ou conteúdo de informações).

### 3.1.1.2 Modelos de processo incremental

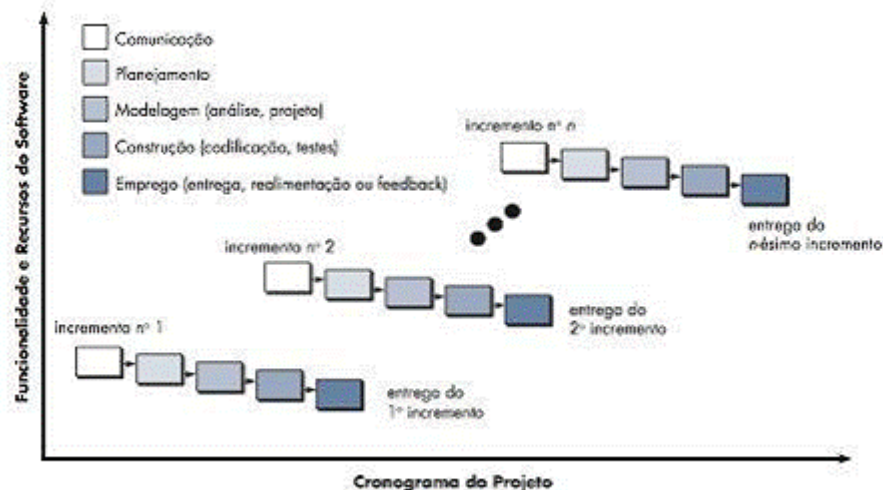
Há situações em que os requisitos iniciais do produto de software pretendido não estão bem definidos e a entrega rápida de um conjunto de funcionalidades ao cliente com o objetivo de refinar e expandir esses requisitos é exigida. Após essa entrega inicial, novas versões de software podem ser elaboradas, chamadas de versões incrementais, cada vez atendendo mais os requisitos, adicionando novas funcionalidades e exercendo um modelo de processo de software incremental (PRESSMAN, 2011, p. 61).

O modelo incremental combina processos lineares com processos paralelos, sendo útil quando houver a necessidade da entrega de um produto ao cliente em um prazo que seja



impossível o desenvolvimento integral das funcionalidades do software. No modelo incremental, serão entregues um ou mais incrementos até a data especificada e o restante do software em incrementos adicionais posteriormente.

Figura 11 - Modelo Incremental



Fonte: PRESSMAN, 2011, p. 61

### 3.1.1.3 Desenvolvimento evolucionário

O desenvolvimento evolucionário consiste no desenvolvimento que lida com mudanças no projeto, com a avaliação periódica do cliente, gerando várias versões do software até a entrega adequada. As ações de especificação, desenvolvimento e validação são integradas, promovendo uma resposta rápida com relação às expectativas do cliente (SOMMERVILLE, 2011, p. 29).

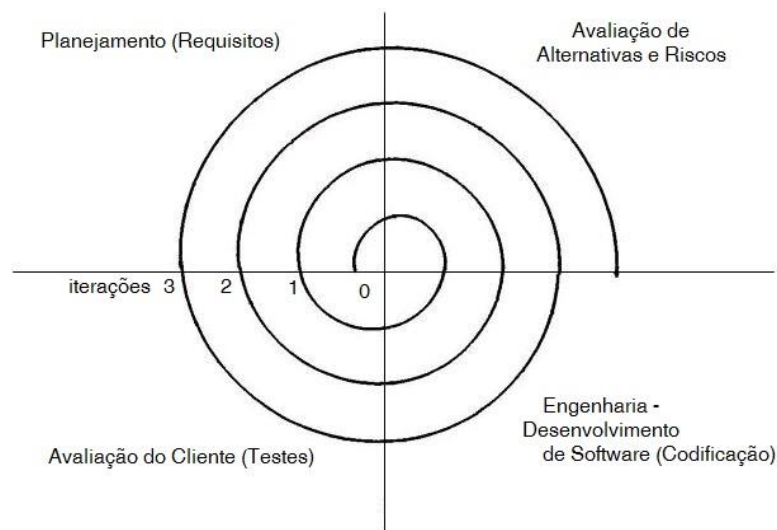
O desenvolvimento evolucionário, além do modelo de processo incremental, pode ser do tipo modelo por **prototipação** (SOMMERVILLE, 2011, p. 30), onde um projeto rápido é desenvolvido contendo aspectos visíveis aos usuários finais do software, dando uma ideia a eles da forma como o sistema navegará entre as telas e ajudando o desenvolvedor nos levantamentos e confirmação de requisitos, testando alternativas de opções de projeto e descobrindo sobre problemas e possíveis soluções. O outro paradigma relacionado ao desenvolvimento evolucionário é o modelo de processo de software em espiral.

#### 3.1.1.4 Modelo espiral de Boehm

O modelo espiral é composto por ilimitadas iterações ou ciclos no desenvolvimento do software. Proposto pelo Dr. Berry Boehm em 1988, o modelo espiral inicia com o **planejamento** do software, onde serão avaliados os objetivos, regras de negócio e requisitos para o desenvolvimento da solução. No segundo quadrante, o ciclo passa pela **avaliação de alternativas e riscos** no desenvolvimento do software, que podem ter origens diversas, dentre elas reações do cliente e falhas no levantamento de requisitos. No terceiro quadrante, estão as **atividades de desenvolvimento do software** propriamente ditas, onde de forma evolutiva, novas funcionalidades são adicionadas à solução através de códigos de linguagem de programação. Por fim, no quarto quadrante, é realizada a **avaliação por parte do cliente** usuário do sistema. Na avaliação por parte do cliente é incorporado os conceitos do modelo de prototipação (TONSIG, 2013, p. 87, grifo nosso).

Todas as atividades no desenvolvimento do software se sobrepõem a cada iteração, e dessa forma cíclica o software evolui até sua forma final.

Figura 12 - Modelo Espiral



Fonte: Elaboração dos autores (2018).

### 3.1.1.5. UML (Linguagem de Modelagem Unificada)

A Linguagem de Modelagem Unificada, em inglês *Unified Modeling Language* (UML) é uma linguagem gráfica que descreve e documenta um projeto de software. A UML foi resultado do esforço de Grady Booch, James Rumbaugh e Ivar Jacobson que resultou na versão 1.0 da UML no início de 1997 (TONSIG, 2013, p. 239).

A UML reuniu diversos artefatos e recursos de métodos orientados a objetos, incorporando um metamodelo que permite a modelagem de diferentes tipos de sistemas desde os mais simples aos sistemas mais complexos, como sistemas distribuídos e concorrentes. O esforço inicial da UML era padronizar um modelo para o desenvolvimento de sistemas e que atendesse o modelo orientado a objetos (TONSIG, 2013, p. 238).

Hoje, a UML provê uma linguagem de modelagem completa aos desenvolvedores de software, possuindo conformidade com *frameworks* e padrões, incentivo a programação orientada a objetos e independência de linguagens de programação e processos de desenvolvimento.

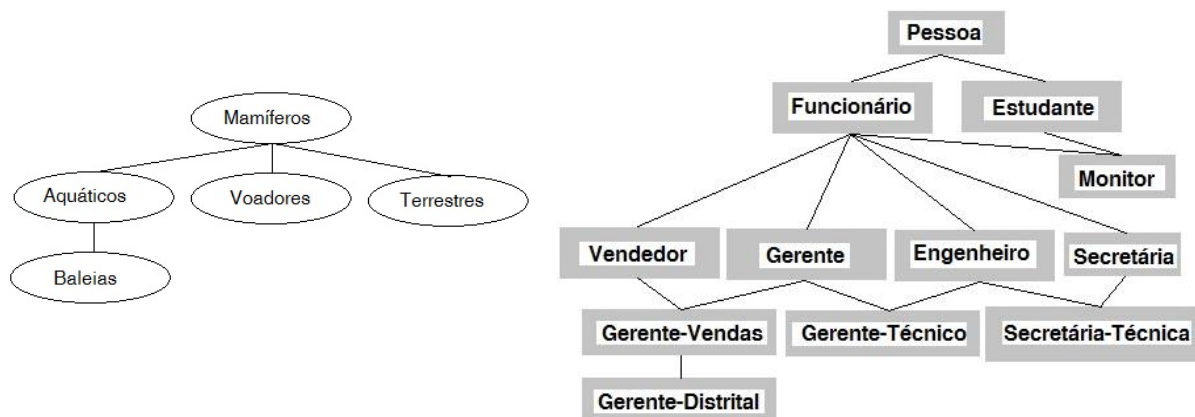
### 3.1.1.6. Paradigma da Orientação a Objetos

O paradigma da orientação a objetos não se fundamenta em princípios novos. A orientação a objetos surgiu no final da década de sessenta com a linguagem SIMULA, sendo aplicado principalmente na linguagem SMALLTALK da Xerox nos anos setenta (TONSIG, 2013, p. 217). Enquanto paradigma significa um modelo ou um padrão, objeto segundo Tonsig (2013, p. 219): “[...] é a representação de elementos físicos do mundo real, que, sob o ponto de vista do problema a ser resolvido, possuem atributos e métodos comuns”.

Os objetos apresentam como propriedades a sua identificação, sua situação ou estados e comportamentos. Objetos são instâncias de uma classe. Portanto, uma classe representa um conjunto de objetos de mesma característica. Uma classe é uma generalização de objetos (TONSIG, 2013, p. 221). Exemplo de classe seria “Veículo” e de objetos da classe “Veículo” seriam os veículos “Gol” e “Corsa”.

As classes representativas de objetos podem herdar características de outras classes, configurando o conceito de herança em orientação a objetos. A herança pode ser múltipla (cada classe herda atributos e comportamentos de várias outras classes) ou simples (cada classe herda atributos e comportamentos de uma única classe).

Figura 13 - Herança Simples (esquerda) e Herança Múltipla



Fonte: Baseado em Tonsig (2013, p. 223).

Os objetos possuem um mecanismo chamado encapsulamento que funciona como uma proteção de acesso aos seus dados (atributos e métodos). Podemos ter atributos e métodos de um objeto que sejam privados (somente podem ser manipulados por métodos internos do objeto), protegidos (somente podem ser manipulados por métodos no mesmo pacote dentro do projeto), bem como atributos e métodos públicos no objeto (qualquer entidade objeto possuirá privilégios para o acesso e alteração).

A característica que cada objeto tem de responder ou agir diferentemente quando acionados da mesma maneira se chama polimorfismo. Um método “Frear” em um objeto do tipo “Veículo” poderá estar implementado como frear até parar (velocidade será zerada), enquanto que em outro “Veículo” esse método diminuirá um pouco a velocidade, por exemplo.

Enfim, o paradigma da orientação a objetos traz como benefícios uma modelagem mais próxima do natural, uma maior fidelidade com o mundo real, uma grande reutilização de classes, levando a projetos mais rápidos, com maior qualidade e trazendo simplicidade à programação e codificação (TONSIG, 2013, p. 230).

### 3.2. REQUISITOS, ANÁLISE E MODELAGEM DO SISTEMA

O requisito em engenharia de software é uma declaração abstrata em alto nível de um serviço ou restrição que um sistema deve oferecer, mas também pode ser uma definição detalhada e formal de uma função do sistema (SOMMERVILLE, 2011, p. 57).

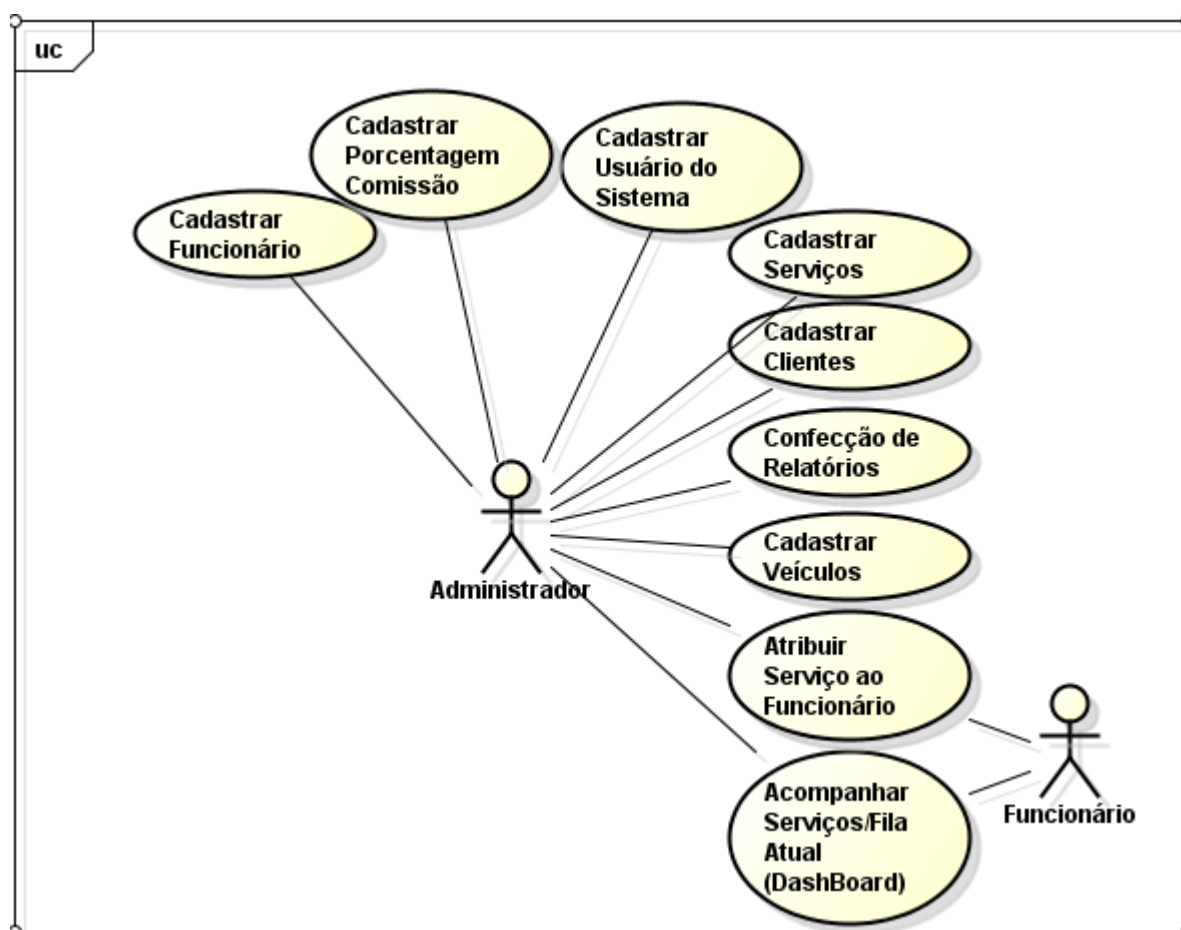
Neste capítulo, serão apresentadas as etapas que compõem requisitos, análise e modelagem do sistema proposto. São elas: diagrama de casos de uso, requisitos funcionais, requisitos não funcionais, protótipos de tela, diagramas de classes e diagrama atividade.

### 3.2.1. Diagrama de casos de uso de alto nível

O diagrama de casos de uso baseado em Linguagem de Modelagem Unificada (UML) é composto pelos atores, que podem ser pessoas ou outros sistemas, pelas classes de interação, que descrevem as ações de interação com o sistema, e são representadas por uma elipse e pelas linhas que fazem a ligação (relacionamento) entre os atores e a interação (SOMMERVILLE, 2011, p. 74).

Sommerville (2011, p. 75) descreve: “[...] casos de uso são técnicas eficazes para elicitar requisitos dos *stakeholders* que vão interagir diretamente com o sistema. Cada tipo de interação pode ser representado como um caso de uso”.

Figura 14 - Diagrama de Casos de Uso do Sistema Web de lava a jato proposto



### 3.2.1.1. Ator Administrador

O ator Administrador terá acesso irrestrito ao sistema, inclusive será responsável por configurar o sistema e acessar todas suas funções.

### 3.2.1.2. Ator Funcionário

O ator Funcionário terá acesso à relação dos clientes e veículos que estão sendo atendidos naquele momento, podendo ser uma relação ou uma lista do dia. Será possível ao Funcionário visualizar ordens de serviço a si atribuídas, bem como antes de finalizado o serviço do cliente, cadastrar-se em alguma etapa de serviço.

## 3.2.2. Requisitos Funcionais

Os requisitos funcionais são especificações dos serviços que o sistema deve oferecer, como deverá ser sua reação a entradas específicas e como será o comportamento do sistema em determinadas situações (SOMMERVILLE, 2011, p. 59).

Sommerville (2011, p. 60) descreve: “Esses requisitos funcionais dos usuários definem os recursos específicos a serem fornecidos pelo sistema. [...] Na prática, para sistemas grandes e complexos, é praticamente impossível alcançar completude e consistência dos requisitos”.

Quadro 2 – Requisitos Funcionais

<b>Cadastrar Funcionário</b> O ator Administrador deverá cadastrar os funcionários do lava a jato.
<b>Cadastrar Porcentagem de Comissão</b> O ator Administrador cadastrará o percentual de comissão aos serviços e etapas de serviços.
<b>Cadastrar Usuário do Sistema</b> O ator Administrador fará a inserção de novos usuário e credenciais de acesso ao sistema.
<b>Cadastrar Serviços</b> O ator Administrador cadastrará os serviços e etapas de serviços que seu estabelecimento disponibilizará ao cliente.

<b>Cadastrar Clientes</b> O ator Administrador fará a inserção de cliente no sistema.
<b>Confecção de Relatórios</b> O ator Administrador inserirá atributos para a produção de relatórios de informação
<b>Cadastrar Veículos</b> O ator Administrador cadastrará os veículos dos clientes que requisitar os serviços.
<b>Atribuir Serviço ao Funcionário</b> O ator administrador atribuirá serviços e etapas de serviço aos funcionários, bem como os atores funcionários poderão se atribuir a determinado serviço.
<b>Acompanhar Serviços/Fila Atual (<i>dashboard</i>)</b> Todos atores terão acesso à fila atual de serviços a serem realizados e serviços já executados recentemente.

Fonte: Elaboração dos autores (2018).

### 3.2.3. Requisitos Não Funcionais

Sobre os requisitos não funcionais, Sommerville (2011, p. 60) afirma: “Os requisitos não funcionais, como o nome sugere, são requisitos que não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários. [...] Sempre que possível, os requisitos não funcionais devem ser escritos quantitativamente, para que possam ser objetivamente testados.”

Realizando uma analogia entre Sommerville e o Modelo de Qualidade da Norma NBR ISO/IEC 9126-1, os requisitos não funcionais possuem os seguintes atributos e podem ser classificados como: Funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.

#### 3.2.3.1. Funcionalidade

- Segurança no acesso, permitindo apenas usuários autorizados ter acesso ao sistema;
- Adequação do produto às necessidades do cliente.

#### 3.2.3.2. Confiabilidade

- Tolerância a falhas, se recuperando de falhas o mais rápido possível.

#### 3.2.3.3. Usabilidade

- Interface simples e organizada.

#### 3.2.3.4. Eficiência

- O sistema deve ser capaz de executar qualquer ação em menos de sete segundos.

#### 3.2.3.5. Manutenibilidade

- O sistema deve ter modularidade na adição de novas funcionalidades, não prejudicando as funções existentes, bem como ser desenvolvido seguindo boas práticas de programação facilitando a manutenção.

#### 3.2.3.6. Portabilidade

- O sistema deve estar preparado para ser acessado de qualquer navegador moderno e de diferentes sistemas operacionais.

### **3.2.4. Protótipos de Tela**

No decorrer desta seção, são apresentados nove protótipos de tela de média fidelidade, definindo a estrutura e *template* básico da interface do sistema proposto.

Sommerville (2011, p. 30) define: “Um protótipo é uma versão inicial de um sistema de software, usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções”.

O sistema será composto inicialmente de uma tela de autenticação, cujos usuários (nome de usuário e senha) estarão pré-cadastrados, com opção de recuperação de senha caso haja esquecimento da mesma por parte do usuário.



Figura 14 - Tela de autenticação do sistema

A imagem mostra uma janela de navegador com o título 'Reis Lavajato'. O endereço da barra de endereços é 'http://127.0.0.1:8080/reislavajato/login'. No centro da tela, há um formulário de login com o título 'Login'. O formulário contém dois campos de entrada: 'Username' e 'Password'. Abaixo dos campos, há dois botões: 'Ok' e 'Limpar', e um link azul 'Esqueci a senha'.

Fonte: Elaboração dos autores (2018).

Após a autenticação do usuário, o sistema redireciona para a tela de *dashboard* mostrada na figura 15. Esta tela exibirá os *menus* de acordo com as autorizações estipuladas para cada tipo de usuário, e no *dashboard* constará uma lista contendo a fila atual de serviços no lava a jato com dados do veículo, do cliente, estágio de atendimento e previsão de término.

Um botão em forma de lupa fornecerá ao administrador informações sobre o andamento dos serviços, como as etapas de serviço realizadas, a se realizar e a relação dos funcionários que executaram essas etapas.

Figura 15 – Tela *dashboard*

Veículo (cliente)	Placa	status	previsão (minutos)
VW Gol	BQC-3280	aguardando!	45
Jair			
Fiat Palio	JGZ-5516	cera	20
Ruben			
Ford Fiesta	PQC-5614	ducha	30
Telmo			
GM Celta	OMO-1115	limpeza interna	10
Ailton			
Citroen C3	BLW-5523	finalizado	0
Maiana			
Hyunday Sonata	BBC-2369	entregue	0
Everton			

Fonte: Elaboração dos autores (2018).

Na aba Cadastros, teremos as telas de cadastro de funcionários, cargos, serviços e etapas de serviços, porcentagem de comissão por serviços, cadastro de clientes e de veículos.

Ao clicar em alguma opção de cadastro, na aba Cadastros, uma lista dos itens salvos no banco de dados do tipo de cadastro escolhido será aberta, contendo as opções Editar (representado por um botão com ícone de lápis), Apagar (representado por um botão com ícone de lixeira) e botão Novo (para salvar um novo item no banco de dados).

A seguir, demonstramos a tela de cadastro de Veículos, bem como a tela de inserção de um novo veículo, tela editar veículo e a tela de exclusão de veículo representada por um *pop-up* de confirmação de exclusão que aparecerá ao usuário.

Figura 16 - Tela Cadastro de Veículos

Placa	Marca/Modelo	Cor	Cliente
BQC-3280	VW Gol	prata	Jair
JGZ-5516	Fiat Palio	cinza	Ruben
PQC-5614	Ford Fiesta	branca	Telmo
OMO-1115	GM Celta	preta	Ailton
BLW-5523	Citroen C3	branca	Maiana
BBC-2369	Hyunday Sonata	branco perolizado	Everton

Novo

Fonte: Elaboração dos Autores (2018).

Figura 17 - Tela de Inserção de Novo Veículo

Novo Veículo

Digite a placa do veículo: xxx-1111

Marca ▼ Modelo ▼ Cor ▼

Entre com o Cliente e contato

Salvar Cancelar

Fonte: Elaboração dos Autores (2018).

Figura 18 - Tela de Edição de Veículo

Reis Lavajato

http://127.0.0.1/lavajato/TeladeCadastrodeVeiculos

	Placa	Marca/Modelo	Cor	Cliente
Dashboard	BQC-3280	VW		Jair
Cadastros	JGZ-5516	Fiat		Ruben
Serviços	PQC-5614	Ford		Telmo
Clientes	OMO-1115	GM		Ailton
Funcionarios	BLW-5523	Citroen		Maiana
Veículos	BBC-2369	Hyundai		Everton
Relatórios				
Configurações				
Sair				

Editar Veículo

Placa do veículo: BQC-3280

VW Gol Prata

Jair 99640-2005

Salvar Cancelar

Novo

Fonte: Elaboração dos Autores (2018).

Figura 19 - Pop-Up de Exclusão de Veículo

Reis Lavajato

http://127.0.0.1/lavajato/TeladeCadastrodeVeiculos

	Placa	Marca/Modelo	Cor	Cliente
Dashboard	BQC-3280	VW Gol	prata	Jair
Cadastros	JGZ-5516	Fiat Palio	cinza	Ruben
Serviços	PQC-5614	Ford Fie		Telmo
Clientes	OMO-1115	GM Cel		Ailton
Funcionarios	BLW-5523	Citroen		Maiana
Veículos	BBC-2369	Hyundai		Everton
Relatórios				
Configurações				
Sair				

Excluir

Deseja excluir o veículo VW Gol, placa BQC-3280, do Jair?

Sim Não

Novo

Fonte: Elaboração dos Autores (2018).

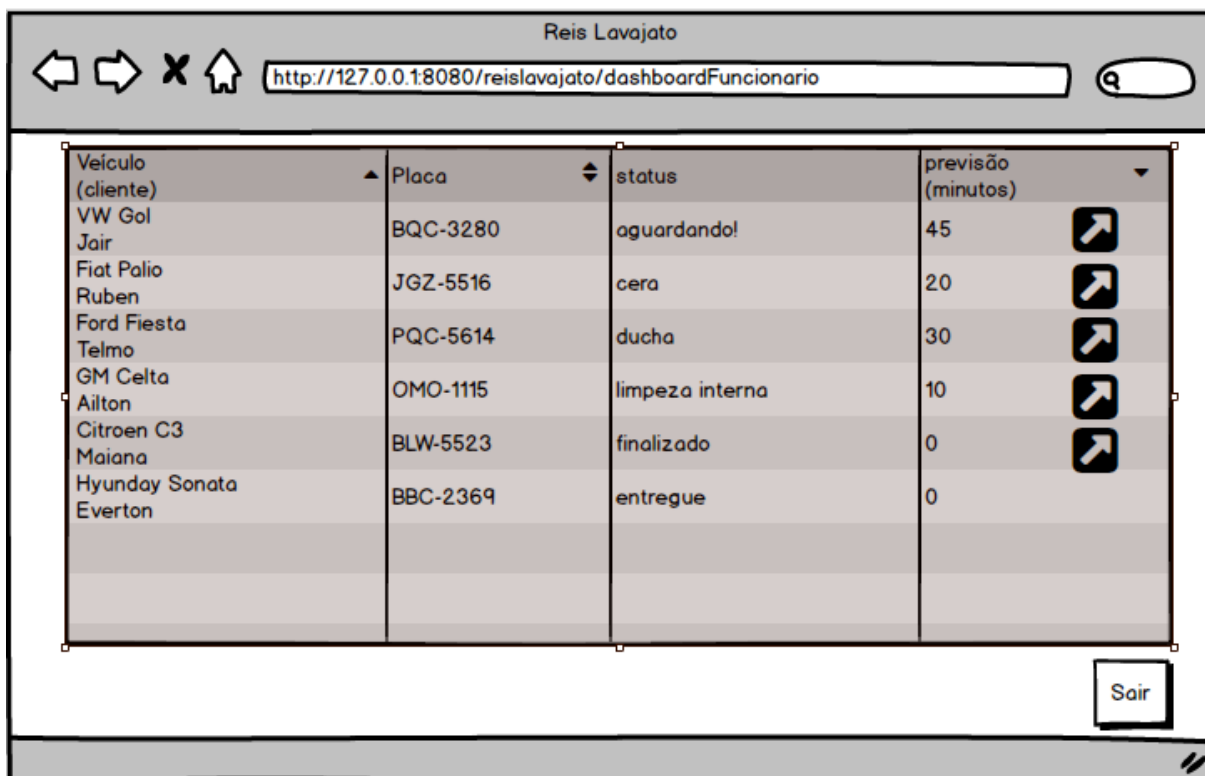
Para o usuário Administrador, a tela de geração de relatórios possibilitará a seleção do período de lançamentos e dos atributos de relatório. Será possível gerar um relatório de cada vez, selecionando o período de tempo e o atributo (relatório de lavagens, de comissão de funcionários, de clientes ou de veículos) e apertando o botão Gerar Relatório.

Figura 20 - Tela de Relatórios para o usuário Administrador

Fonte: Elaboração dos Autores (2018).

Uma opção de tela/*dashboard* para os funcionários que trabalham no lava a jato foi proposta, contendo uma lista com a fila dos veículos e serviços atuais do lava a jato. Além disso, essa *view* deverá redirecionar a outra *view* de cadastro de vínculo de serviços executados pelo próprio funcionário em determinado veículo ou cliente.

Um requisito dessa tela de cadastro de vínculo de serviço e funcionário são botões e textos com fontes e tamanho relativamente grandes, possibilitando uma maior facilidade ao funcionário que utiliza luvas, óculos de proteção ou outro equipamento de proteção, selecionar o veículo e registrar-se como executor da etapa de serviço.

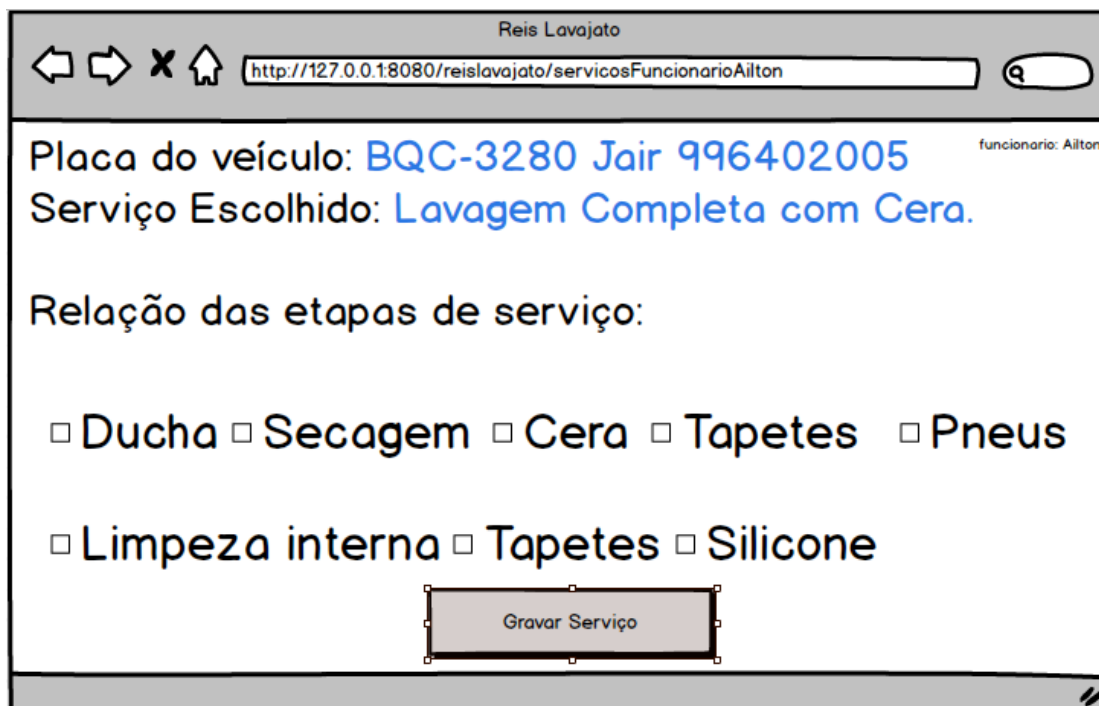
Figura 21 - Tela de *dashboard* para o usuário perfil Funcionário


Veículo (cliente)	Placa	status	previsão (minutos)
VW Gol	BQC-3280	aguardando!	45
Fiat Palio	JGZ-5516	cera	20
Ruben	PQC-5614	ducha	30
Ford Fiesta	OMO-1115	limpeza interna	10
Telmo	BLW-5523	finalizado	0
GM Celta	BBC-2369	entregue	0
Ailton			
Citroen C3			
Maiana			
Hyunday Sonata			
Everton			

Sair

Fonte: Elaborado pelos autores (2018).

Figura 22 - Tela de Cadastro de Vínculo de Serviço e Funcionário



Reis Lavajato

http://127.0.0.1:8080/reislavajato/servicosFuncionarioAilton

Placa do veículo: **BQC-3280** **Jair** 996402005 funcionario: Ailton

Serviço Escolhido: **Lavagem Completa com Cera.**

Relação das etapas de serviço:

☐ Ducha ☐ Secagem ☐ Cera ☐ Tapetes ☐ Pneus

☐ Limpeza interna ☐ Tapetes ☐ Silicone

Gravar Serviço

Fonte: Elaborado pelos autores (2018).



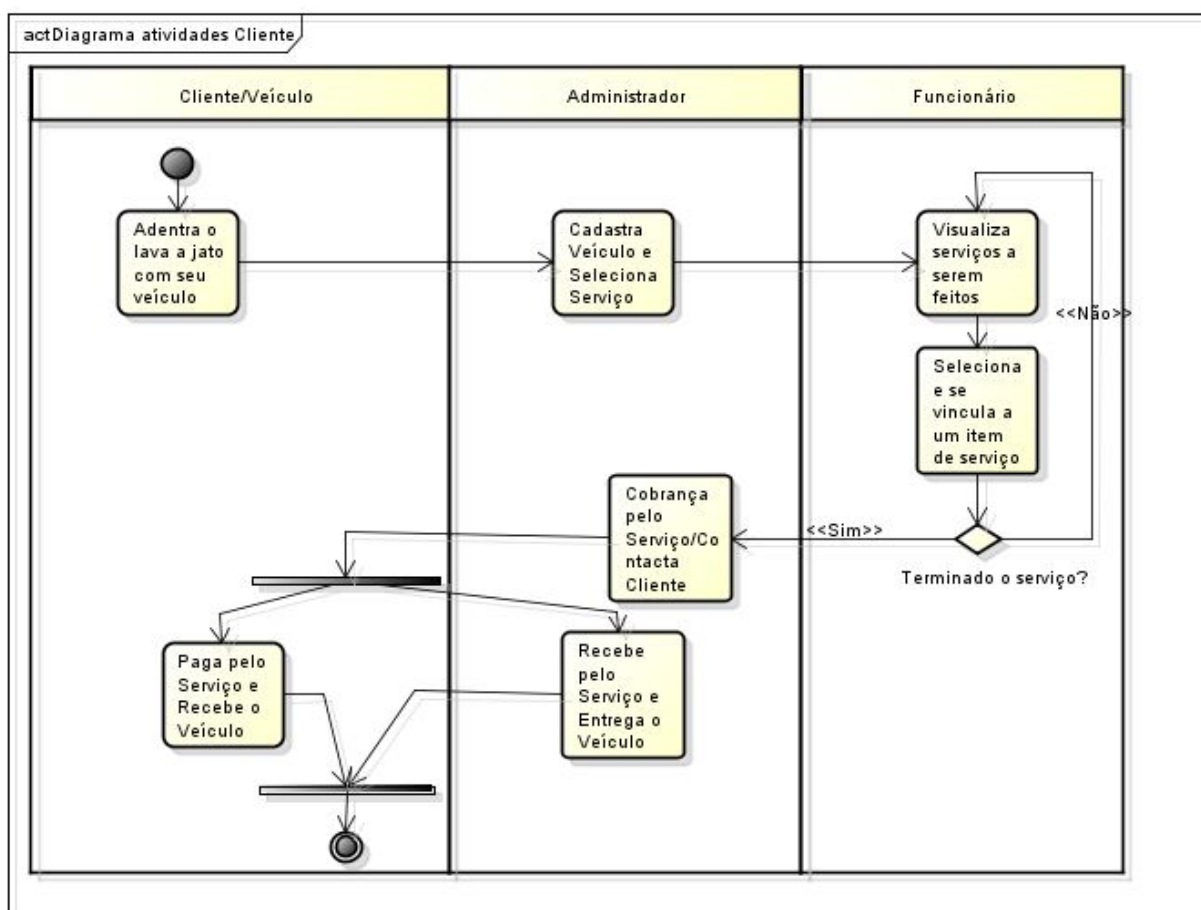
O diagrama de classes também é construído no padrão e em conformidade com a Linguagem de Modelagem Unificada (UML).

### 3.2.6. Diagrama de Atividade

Pressman (2011, p. 737) descreve: “O diagrama de atividade mostra o comportamento dinâmico de um sistema ou parte de um sistema através do fluxo de controle entre ações que o sistema executa. Ele é similar a um fluxograma exceto que pode mostrar fluxos concorrentes”.

O diagrama de atividade do sistema proposto neste projeto foi construído do momento da chegada do cliente ao lava a jato, passando pelo seu cadastro e do veículo pelo administrador, realização dos serviços pelos funcionários até a entrega final do veículo ao cliente.

Figura 24 - Diagrama de Atividade



Fonte: Elaboração dos autores (2018).



## 4. ESPECIFICAÇÃO DAS FERRAMENTAS PARA O DESENVOLVIMENTO SISTEMA WEB PROPOSTO

Neste capítulo, exporemos as ferramentas utilizadas na engenharia desse sistema, bem como suas respectivas fases, desde a etapa de modelagem até a implementação do sistema.

O modelo proposto para o projeto, modelo cliente-servidor, é composto por um ou mais clientes que realizam as requisições ao servidor. Essas requisições serão feitas pelo navegador, que faz o papel de cliente, gerando uma série de requisições que serão tratadas, gerenciadas e então retornadas ao navegador, demonstrando o resultado desse processo de forma amigável ao usuário.

A escolha das ferramentas para o desenvolvimento desse trabalho levou em conta o conhecimento técnico dos autores, a facilidade oferecida, produtividade e flexibilidade na busca dos objetivos. Foram ferramentas selecionadas que melhor se adequavam à produção de plataforma Web, livres, produtivas e de uso significativo no mercado.

Serão apresentadas brevemente as ferramentas: Balsamic Mockup, Astah, Java, JSF (PrimeFaces), JPA (Hibernate), Eclipse IDE (JBoss Tools, Maven), SpringMVC, MySQL (dbForge ou Workbench), Apache TomCat, JUnit, Jasper Reports, RiouxSVN, Trello.

### 4.1. BALSAMIC MOCKUP

O Balsamic Mockups é uma ferramenta da Balsamic Studio de prototipagem de tela rápida e intuitiva. Também chamada de ferramenta de *wireframing*, o Balsamic Mockups reproduz a experiência de esboçar os protótipos em um quadro branco, porém de forma digital, em um computador. Pela facilidade de uso e velocidade de produção, os esboços podem ser avaliados e alterados de maneira imediata (MOCKUPS, 2017). Foi utilizada uma versão *trial* dessa ferramenta, visto se tratar de software que é pago.

### 4.2. ASTAH COMMUNITY

O Astah da Change Vision, Inc na sua versão Community é um software gratuito para modelagem UML versão 2 (Unified Modeling Language – Linguagem de Modelagem Unificada). A ferramenta na sua versão gratuita possui suporte para a confecção de onze tipos de diagramas UML, dentre eles o diagrama de classes, caso de uso, sequência, atividades,

comunicação, máquina de estados, componentes, implantação, estruturas compostas, de objetos, de pacotes e tabela de transição de estados. É uma ferramenta simples, leve e de fácil utilização (ASTAH, 2017).

#### 4.3. JAVA

Utilizaremos a linguagem de programação Java neste trabalho pelo conhecimento prévio da linguagem por parte dos autores, bem como pelas suas qualidades e características. Java é uma linguagem de programação lançada oficialmente pela empresa *Sun Microsystems* em 1995 e implementa o paradigma de orientação a objetos, *multithreading* e gerenciamento de memória automático. Java é uma linguagem poderosa, cujas ferramentas são gratuitas e presente em uma variedade imensa de dispositivos informáticos (JAVA, 2017).

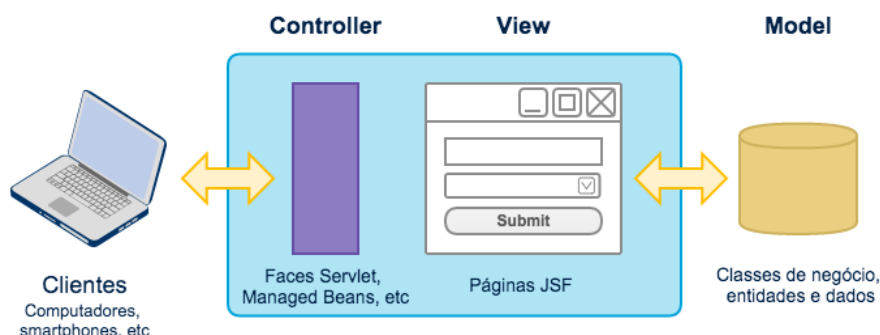
O Java é a base para praticamente todos os tipos de aplicações em rede e é o padrão global para o desenvolvimento e distribuição de aplicações móveis e incorporadas, jogos, conteúdo baseado na Web e softwares corporativos. Com mais de 9 milhões de desenvolvedores em todo o mundo, de forma eficiente, o Java permite que você desenvolva, implante e use aplicações e serviços estimulantes. (JAVA, 2017).

Utilizaremos a última versão (8u161) da plataforma Java, *Standard Edition* (SE) *Development Kit* (JDK) disponibilizada em janeiro de 2018.

#### 4.4. JAVA SERVER FACES (JSF)

O JSF é um framework (K19, 2013 p. 77) que nos permite criar aplicações web em Java utilizando componentes visuais pré-prontos que são renderizados e exibidos em formato *HiperText Markup Language* (HTML) no navegador (CAELUM, FJ-22, 2013, p. 81). JSF é fortemente baseado nos padrões MVC (*model-view-controller*), separando as camadas de apresentação (visualização) das camadas de aplicação (controle e modelo) (CAELUM, FJ-22, 2013, p. 82).

Em JSF, o controle é feito através de uma servlet chamada Faces Servlet, opcionalmente, por arquivos XML de configuração e por vários manipuladores de ações e observadores de eventos. A Faces Servlet recebe as requisições dos usuários na web, redireciona para o modelo e envia uma resposta. (FARIA, 2015, p. 51).

Figura 25 - Padrão *Model-View-Controller* (MVC)

Fonte: FARIA, 2015, p. 52.

Enquanto o JSF é uma especificação, existem bibliotecas criadas por organizações que trabalham na criação de componentes personalizados e sofisticados, que vão muito além da especificação. Exemplos dessas bibliotecas são PrimeFaces (usada nesse projeto na versão 2.2.14), RichFaces e IceFaces (CAELUM, FJ-22, 2013, p. 83). Cada biblioteca oferece *showcases* dos componentes de tela nas suas respectivas *web pages*.

Figura 26 - Bibliotecas de componentes de JSF



Fonte: CAELUM, FJ-22, 2013, p. 83.

#### 4.5. JPA (*JAVA PERSISTENCE API*)

O *Java Persistence API* (JPA) surgiu, dentre outros motivos, como uma forma de diminuir o tempo no desenvolvimento de aplicações na linguagem Java em ambientes corporativos, mais especificamente reduzir o tempo gasto com codificações de consultas em Linguagem de Consulta Estruturada (*Structured Query Language* - SQL) e códigos *Java DataBase Connectivity* (JDBC). Outros motivos importantes estão no fato de cada banco de dados possuir suas peculiaridades na linguagem SQL, mesmo ela sendo uma linguagem padrão ANSI (*American National Standards Institute*), além de sua conformidade com o paradigma de programação orientado a objetos quando desenvolvemos em uma aplicação Java.

A programação orientada a objetos difere muito do esquema entidade relacional e precisamos pensar das duas maneiras para fazer um único sistema. [...] a todo momento devemos “transformar” objetos em registros e registros em objetos. (CAELUM, FJ-21, p. 199).

Caelum explica mais sobre a origem e características do *Hibernate* na citação abaixo.

Ferramentas para auxiliar nesta tarefa tornaram-se populares entre os desenvolvedores Java e são conhecidas como ferramentas de Mapeamento Objeto - Relacional (ORM). O *Hibernate* é uma ferramenta ORM *open source* e é a líder de mercado, sendo a inspiração para a especificação Java Persistence API (JPA). O *Hibernate* nasceu sem JPA mas hoje em dia é comum acessar o *Hibernate* pela especificação JPA. (CAELUM, FJ-21, p. 200).

Apesar de termos outras implementações do JPA, como *EclipseLink* da *Eclipse Foundation* e *OpenJPA* da Apache, utilizamos o *Hibernate* da *JBoss* neste trabalho.

#### 4.6. ECLIPSE IDE

O Eclipse é um ambiente integrado de desenvolvimento (*Integrated Development Enterprise – IDE*), altamente útil e eficiente, de código-fonte aberto e considerado uma plataforma poderosíssima por sua capacidade de desenvolvimento de *plug-ins* (HEMRAJANI, 2007, p. 138).

O Eclipse foi originalmente desenvolvido pela *Object Technology International* (OTI), que mais tarde foi comprada pela empresa norte-americana IBM. Em novembro de 2001, a IBM doou a tecnologia do Eclipse, avaliada em dezenas de milhões de dólares na época, a organizações de código-fonte aberto e recrutou vários membros para conjuntamente, desenvolverem produtos altamente integrados para essa plataforma (na forma de *plug-ins*). O número de membros da Fundação Eclipse vem crescendo rapidamente, contando com um seleto grupo, dentre eles: IBM, Google, Red Hat, GitHub, Huawei, Mercedes-Bens, dentre outros (HEMRAJANI, 2007, p. 139).

#### 4.7. SPRING FRAMEWORK

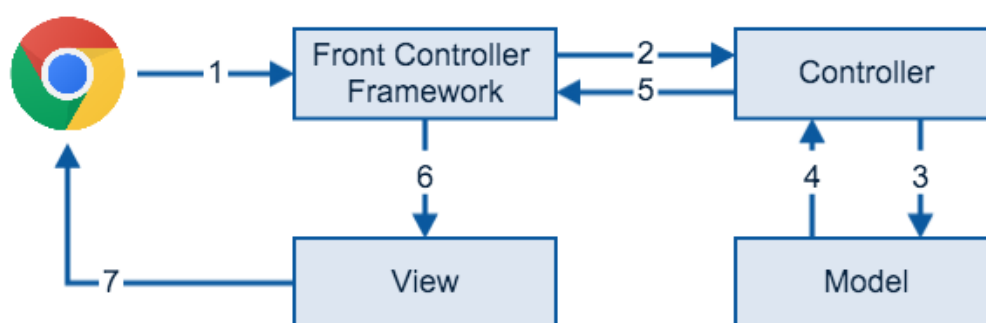
O Spring é composto por vinte e três projetos implementados, dois projetos comunitários e quatro projetos para o futuro, criados para ajudar a construir e configurar aplicativos corporativos modernos baseados em Java (SPRING PROJECTS, 2017). Dentre esses projetos, utilizaremos o Spring *Framework*, onde está incluído o Spring Web MVC que

é a estrutura web original do projeto. O Spring Web MVC é conceituado por Afonso (2017a, p. 15) na citação abaixo.

[...] é o framework que te ajuda no desenvolvimento de aplicações web robustas, flexíveis e com uma clara separação de responsabilidades nos papéis do tratamento da requisição.

MVC é acrônimo de Model, View e Controller, e entender bem o que cada um deve fazer na aplicação é importante para termos uma aplicação bem escrita e fácil para dar manutenção. (AFONSO, 2017a, p. 15).

Figura 27 - Spring Web MVC



Fonte: AFONSO, 2017a, p. 16.

As solicitações HTTP (*HyperText Transfer Protocol*) oriundas do navegador são recebidas pelo servidor Apache Tomcat que roda a aplicação web com Spring MVC. Essa solicitação é recebida no controlador do *framework* Spring MVC, que procurará a classe que fará o papel de *controller* e tratará os dados enviados do *browser*. O *controller* passará os dados ao *model* que executará as regras de negócio, validações, processamento e acesso ao banco de dados. O resultado dessas operações retornam ao *controller* que encaminha os dados à respectiva *view*, que transforma os dados, renderizando e entregando um código HTML ao navegador (AFONSO, 2017a, p. 16 e HEMRAJANI, 2007, p. 109).

O framework Spring Web MVC trabalha com injeção de dependências (*Dependency Injection – DI*), que é um tipo de inversão de controle (*Inversion of Control – IoC*) sendo um mecanismo de instanciar classes que um objeto necessita para funcionar, conforme Afonso (2017a, p. 14).

[...] inversão de controle [...] que dá nome ao processo de prover instâncias de classes que um objeto precisa para funcionar.

A grande vantagem desse conceito é que nós conseguimos programar voltados para interfaces e, com isso, manter o baixo acoplamento entre as classes de um mesmo projeto.

Com certeza, essa característica é uma grande vantagem para a arquitetura do seu sistema, assim como é para o próprio Spring. Como foi dito mais no início, essa funcionalidade é a base de todo o ecossistema Spring. É difícil pensar em Spring sem a injeção de dependências. (AFONSO, 2017a, p. 14).

A anotação *@Autowired* avisa ao Spring *Framework* onde são os pontos injetáveis em uma classe. Essa anotação pode ser utilizada em atributos, construtores ou métodos (normalmente os *setters*).

Para que uma instância possa ser injetada em algum dos pontos de injeção, é necessário que ela se torne um *bean Spring*, com a anotação *@Component* ou com alguma de suas especializações (AFONSO, 2017b):

- *@Repository*: usada para casos de persistência;
- *@Service*: usada em componentes específicos de serviço;
- *@Controller*: usada em componentes de controlador.

Alguns benefícios do Spring Web MVC são as facilidades de testes, a separação clara de funções, a flexibilidade com tecnologias de *front-end* e o ambiente mais leve, bem projetado, flexível e robusto (HEMRAJANI, 2007, p. 108).

#### 4.8. MYSQL SERVER

O banco de dados que será utilizado em nosso projeto será o *MySQL Server* e utilizaremos o ambiente de ferramentas integradas oficial do MySQL, o *MySQL Workbench*. O *MySQL Server* é uma versão de *download* livre do banco de dados de código aberto mais popular do mundo, que é suportado por uma comunidade ativa de desenvolvedores e entusiastas de código aberto (MYSQL, 2017).

O *MySQL Workbench* é uma ferramenta que oferece aos desenvolvedores e administradores de bancos de dados, entre outras funcionalidades, modelagem e *design* de banco de dados, desenvolvimento SQL, administração de banco de dados e migração de banco de dados (MYSQL, 2017).

#### 4.9. APACHE TOMCAT

Apache Tomcat é uma implementação de código-fonte aberto, colaborativa (semelhante ao *Eclipse Foundation*) e participativa de um *Web Container* (APACHE

TOMCAT, 2017). As páginas JSF (*Java Server Faces*), JSP (*Java Server Pages*), arquivos estáticos (HTML, CSS, imagens) e os *Servlets* necessitam de um *Web Container* para serem executados.

Um Web Container é responsável:

- Pelo envio e recebimento de mensagens HTTP.
- Por permitir que as aplicações sejam acessadas simultaneamente por vários usuários de uma maneira eficiente.
- Por permitir que as páginas de uma aplicação web sejam geradas dinamicamente. (K19, 2013, p. 61).

Além de ser muito popular, usaremos o *Web Container* Apache Tomcat por ser gratuito e leve (AFONSO, 2017, p. 18).

#### 4.10. JUNIT

Testes de unidade são testes que testam apenas classes ou métodos, verificando se seus comportamentos estão dentro do desejado (CAELUM, FJ-22, p. 34).

O JUnit (junit.org) é um *framework* de teste do Java, construído em código-fonte aberto, muito utilizado nos testes de unidade do código Java. Originalmente, o JUnit foi escrito por Erich Gamma, integrante da *Gang of Four* do livro *Design Patterns* e Kent Beck, escritor da *Extreme Programming* (HEMRAJANI, 2007, p. 56).

O desenvolvimento baseado em testes referenciado por Kent Beck, chamado de TDD (*test-driven development*), pode trazer benefícios ao projeto, como um código melhor, mais limpo e eficiente (HEMRAJANI, 2007, p. 56).

#### 4.11. JASPER REPORTS

O JasperReports é uma ferramenta baseada no Eclipse IDE, inteiramente escrita em Java, de código aberto, para produção de relatórios. O JasperReports gera relatórios de alta qualidade em diversos formatos, dentre eles, RTF, PDF, CSV, JSON, arquivos de texto (.doc ou .odt), planilha eletrônica ou XML (JASPERSOFT, 2017).

#### 4.12. RIOUXSVN

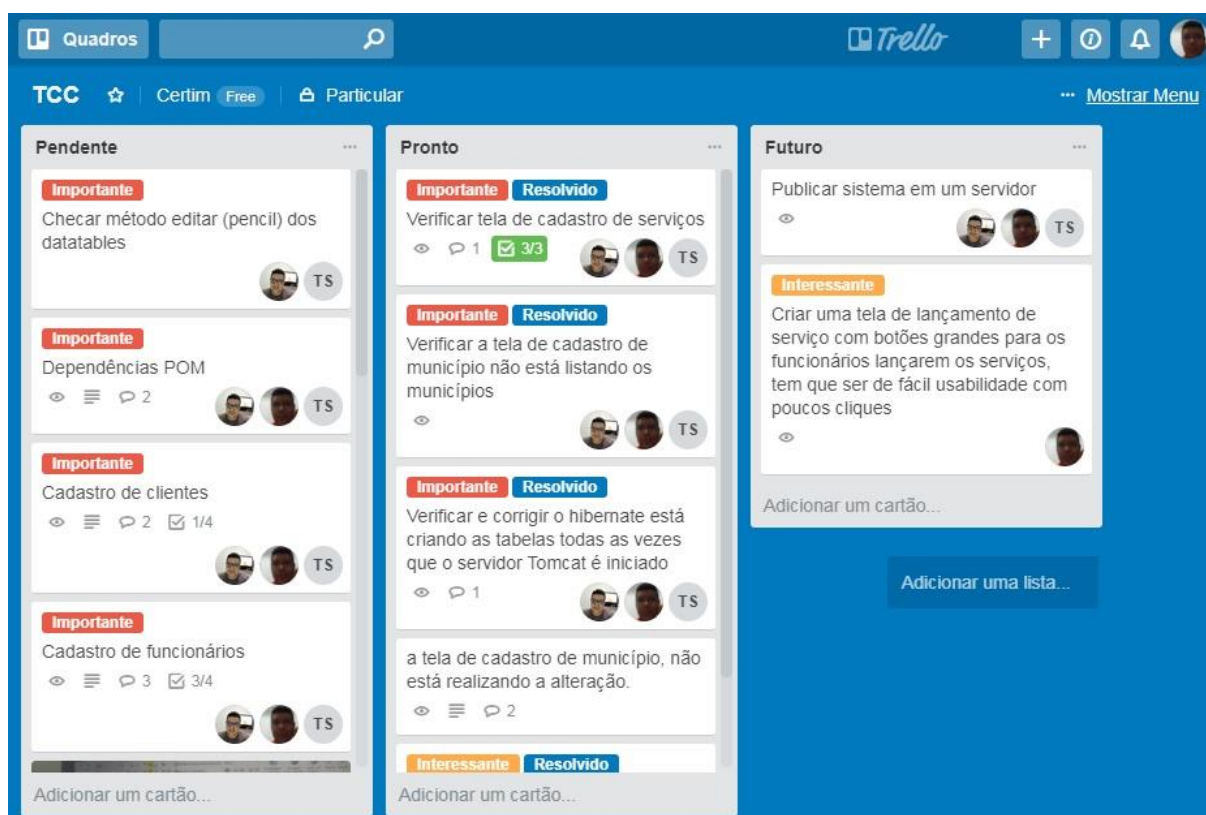
O RiouxSVN é um sistema de controle de versões de projetos Java baseado na Web. Ele é um serviço de hospedagem gratuito, privado e de qualidade. Com RiouxSVN temos 50 MegaBytes de armazenamento no repositório, podemos compartilhar o projeto com um número ilimitado de usuários, além de não possui anúncios comerciais (RIOUXSVN, 2017).

Na comunicação e sincronização do RiouxSVN com o Eclipse IDE, foi utilizado o *plug-in* Subclipse 4.2.3.

#### 4.13. TRELLO

O Trello é uma ferramenta de gerenciamento e organização de projetos baseada na Web. Milhões de pessoas no mundo utilizam o Trello nas mais variadas tarefas (TRELLO, 2017). Essa ferramenta chamou a atenção dos autores pela facilidade de uso, gratuidade e flexibilidade. Grandes e reconhecidas organizações, como a Google e a RedHat utilizam o Trello na organização de seus projetos.

Figura 28 - Gerenciamento do Projeto com o Trello



Fonte: PrintScreen do sistema web Trello



## IMPLEMENTAÇÃO / SISTEMA DESENVOLVIDO

RESUMO DAS ETAPAS, DIFICULDADES;  
ENTIDADES, TELAS+CODIGOS, CONTROLLER+CODIGOS  
(MANAGEDBEAN), MODEL+CODIGOS (DAO, GENERIC DAO);

## 5 IMPLEMENTAÇÃO E DESENVOLVIMENTO DO SISTEMA

O conjunto das atividades envolvidas no processo de desenvolvimento do Sistema Web para o Controle e Gerenciamento de Serviços de Lavagem de Veículos, se constituíram de etapas, sendo elas: modelagem do sistema, definições de arquitetura e projeto, construção da interface e implementação do código-fonte com testes de unidade.

Na **Análise da Solução** (Capítulo 3), fora executada a modelagem do sistema e casos de uso foram criados para explicitar as atividades e funções de cada ator. Fora observado que se um diagrama de caso de uso fosse mal executado, o sistema poderia tomar uma extensão que, por conta do tempo, jamais poderia ser entregue. Ainda nesta etapa, foram definidos os requisitos funcionais, especificando as funcionalidades que o sistema deve oferecer, e os requisitos não funcionais, delimitando restrições e atributos de qualidade do sistema. Protótipos de tela foram criados para certificar a consonância dos requisitos funcionais com a interface do usuário. O diagrama de classes foi apresentado demonstrando o esboço das entidades (classes) e seus relacionamentos, além do diagrama de atividade demonstrando o fluxo do processo no sistema proposto.

Na etapa de escolha das **Ferramentas para o Desenvolvimento do Sistema Proposto** (Capítulo 4), foram definidos cronogramas, linguagem de programação, ferramentas e *frameworks* a serem utilizados no desenvolvimento do sistema. No decorrer do desenvolvimento, foram agregadas soluções comumente utilizadas no mercado, bem como soluções que se fizeram necessárias ao projeto.

A etapa de **Construção da Interface** foi executada concomitantemente com a etapa descrita à seguir, de desenvolvimento do código-fonte, e tomamos como base os protótipos de tela criados na etapa de modelagem do sistema. As interfaces de usuário foram construídas utilizando-se de um *template*, sendo possível o desenvolvimento de uma interface iterativa, intuitiva e com *design* elegante.

A **Implementação do Código-Fonte** (este Capítulo) foi a etapa que mais tomou tempo entre as demais etapas do projeto. O sistema foi sendo desenvolvido de acordo com o prazo, e ideias menos ambiciosas tiveram que ser colocadas em prática. Por outro lado, ideias mais ambiciosas foram lançadas no Capítulo 7 como trabalhos futuros.

Mesclar a parte de *front-end* e *back-end* poderia ser uma das atividades mais difíceis desta etapa, porém vivenciou-se uma atividade tranquila e com pequenos desafios. O grande desafio na verdade foi implantar as regras do sistema em si, bem como a implementação dos relacionamentos entre as entidades, *enums* e fatos, que levou à criação de novas tabelas no banco de dados em alguns casos.

## **IMPLANTAÇÃO E AVALIAÇÃO DO SISTEMA (se der tempo!)**

QUESTIONÁRIO AVALIATIVO PELO CLIENTE;

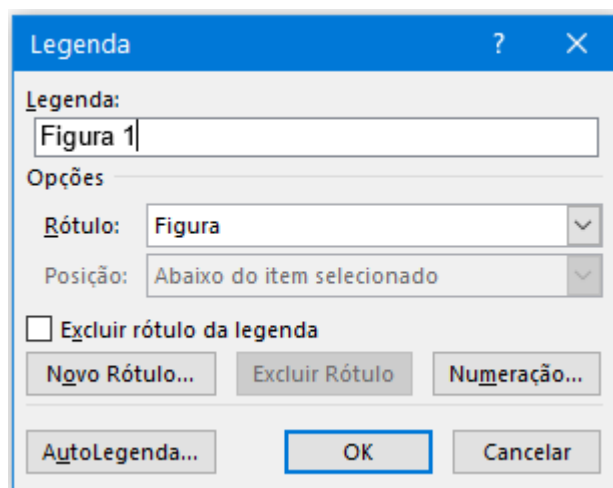
## **CONCLUSÕES E TRABALHOS FUTUROS**

## **REFERÊNCIAS**

#### 4.14. INSERINDO LEGENDA DE FIGURAS

Utilize a Aba “Referências” para colocar legendas nas figuras e tabelas, clicando em “Inserir Legenda” (Figura 1). A formatação adequada das legendas também já está definida nos estilos.

Figura 1 - Janela de legenda das imagens



Fonte: Autor

#### 4.15. INSERINDO NORMA ABNT NAS REFERÊNCIAS

A partir da versão 2007 o Microsoft Word conta com um Gerenciador de referências que pode ser encontrada na Guia Referências. Esse Gerenciador de Referências já vem com várias referências pré-instaladas, infelizmente, as Norma ABNT não faz parte desta lista, porém, podemos adicioná-la como veremos neste tutorial.

Para fazer uso do gerenciador de referências bibliográficas, é necessário incluir o estilo correspondente as normas da ABNT. Sendo assim, o primeiro passo é baixar o arquivo deste estilo.

Acesse o site <http://bibword.codeplex.com/releases/view/27212> para fazer o download do estilo **ABNT NBR 6023:2002\***

Depois que baixou o arquivo do estilo ABNT desejado, só resta copiá-lo para a pasta correta onde o Word 2010 armazena seus estilos.

Se o Windows for até a versão 7, a pasta dos estilos o local é em:

- C:\Program Files (x86)\Microsoft Office\Office14\Bibliography\Style  
ou
- C:\Arquivos de Programas (x86)\Microsoft Office\Office14\Bibliography\Style

Versões mais recentes deverão ser adicionadas no local:

- C:\Users\“nome\_do\_seu\_usuario”\AppData\Roaming\Microsoft\Bibliography\Style

## REFERÊNCIAS

ABNT, Associação Brasileira de Normas Técnicas. **NBR ISO/IEC 9126-1. Engenharia de software – Qualidade de produto.** Parte 1: Modelo de Qualidade. Rio de Janeiro: ABNT – Associação Brasileira de Normas Técnicas, jun. 2003. Disponível em: < [https://aplicacoes.mds.gov.br/sagirms/simulacao/sum\\_executivo/pdf/fichatecnica\\_21.pdf](https://aplicacoes.mds.gov.br/sagirms/simulacao/sum_executivo/pdf/fichatecnica_21.pdf)>.

Acesso em: 01 fev. 2018.

\_\_\_\_\_. **NBR ISO/IEC 6023. Informação e documentação – Referências - Elaboração.** Rio de Janeiro: ABNT – Associação Brasileira de Normas Técnicas, ago. 2002. Disponível em: < <http://www.justicaeleitoral.jus.br/arquivos/tse-norma-abnt-6023>>. Acesso em: 01 fev. 2018.

AFONSO, Alexandre. **Produtividade no Desenvolvimento de Aplicações Web com Spring Boot.** 3. ed. São Paulo: AlgaWorks Softwares, Treinamentos e Serviços Ltda., 2017a.

\_\_\_\_\_. **Injeção de Dependências com Spring.** São Paulo: AlgaWorks Softwares, Treinamentos e Serviços Ltda., 2017b. Disponível em: < <http://blog.algaworks.com/injecao-de-dependencias-com-spring/>>. Acesso em: 22 out. 2017.

APACHE TOMCAT. **Página Oficial do Apache Tomcat.** Disponível em: <<http://tomcat.apache.org/index.html>>. Acesso em: 18 out. 2017.

ASTAH. **Página oficial do Astah.** Change Vision, Inc. Disponível em: <<http://astah.net>>. Acesso em 04 nov. 2017.

BRAGG, Steven. **How to calculate a commission.** 18 dez. 2017. Disponível em: <https://www.accountingtools.com/articles/how-to-calculate-a-commission.html>. Acesso em: 04 set. 2017.

CAELUM, Curso FJ-22. **Apostila Curso FJ-22: Laboratório Java com Testes, JSF e Design Patterns.** Versão: 18.3.13. Grupo Caelum: São Paulo. Disponível em: <[www.caelum.com.br/apostilas](http://www.caelum.com.br/apostilas)>. Acesso em: 01 nov. 2017.

CAELUM, Curso FJ-21. **Apostila Curso FJ-21: Java para Desenvolvimento Web.** Versão: 17.0.6. Grupo Caelum: São Paulo. Disponível em: <[www.caelum.com.br/apostilas](http://www.caelum.com.br/apostilas)>. Acesso em: 03 nov. 2017.

CHAMOUN, Roberto. **Idéias de Negócios: Como montar um lava-jato.** Sebrae, 2012. Disponível em: <http://www.sebrae.com.br/sites/PortalSebrae/ideias/Como-montar-um-lava%E2%80%93jato>. Acesso em: 28 ago. 2017.

COMISSÃO. In: DICIONÁRIO Online de Português. Brasil: Dicio, 2009-2018. Disponível em: <<https://www.dicio.com.br>>. Acesso em: 04 set. 2017.

DEITEL, P. J.. **Java: Como programar.** 8.ed. São Paulo: Pearson Prentice Hall, 2010.

FARIA, Thiago. **Java EE7 Com JSF, PrimeFaces e CDI.** 2. ed. São Paulo: AlgaWorks Softwares, Treinamentos e Serviços Ltda., 2015.

HEMRAJANI, Anil. **Desenvolvimento ágil em Java com Spring, Hibernate e Eclipse.** São Paulo: Pearson Prentice Hall, 2007.

JASPERSOFT Community. **Página oficial do JasperReports.** TIBCO Software Inc [US]. Disponível em: <<https://community.jaspersoft.com/>>. Acesso em 04 nov. 2017.

JAVA. **Página oficial do Java.** Oracle Corporation [US]. Disponível em: <[https://java.com/pt\\_BR/download/faq/whatis\\_java.xml](https://java.com/pt_BR/download/faq/whatis_java.xml)>. Acesso em 04 nov. 2017.

K19. K19 Treinamentos. **Desenvolvimento Web com JSF2 e JPA2.** K19 Treinamentos: São Paulo, 20 fev. 2013. Disponível em: <<http://www.dai.ifma.edu.br/~mlcsilva/aulasdsweb/Material1.pdf>>. Acesso em: 01 nov. 2017.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Fundamentos de metodologia científica**. 7.ed. São Paulo: Atlas, 2010.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Técnica de pesquisa**. 7.ed. São Paulo: Atlas, 2008.

MOCKUPS, Balsamiq. **Página oficial da Balsamic Studios**. 2008-2017. Disponível em: < <https://balsamiq.com>>. Acesso em 04 nov. 2017.

MYSQL. **Página Oficial do MySQL**. Disponível em: < <https://dev.mysql.com/downloads/mysql/>>. Acesso em: 19 out. 2017.

OBSERVATÓRIO DAS METRÓPOLES. **Evolução da frota de automóveis e motos no Brasil 2001-2012 (Relatório 2013)**. Rio de Janeiro, Outubro, 2013. Disponível em: < [http://www.observatoriodasmetropoles.net/download/auto\\_motos2013.pdf](http://www.observatoriodasmetropoles.net/download/auto_motos2013.pdf)>. Acesso em 28 ago. 2017.

PRESSMAN, Roger S.. **Engenharia de Software: Uma Abordagem Profissional**. 7. ed. Porto Alegre: AMGH, 2011.

RIOXSVN. **Página Oficial do RiouxSVN**. Disponível em: < <https://riouxsvn.com/>>. Acesso em: 28 out. 2017.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SPRING PROJECTS. **Página Oficial dos projetos do Spring**. Disponível em: < <https://spring.io/projects>>. Acesso em: 20 out. 2017.

TONSIG, Sérgio Luiz. **Engenharia de Software: Análise e Projeto de Sistemas**. 2. Ed. Rio de Janeiro: Editora Ciência Moderna Ltda., 2013.

TRELLO. **Página oficial do Trello**. Disponível em: < <https://trello.com/>>. Acesso em 04 nov. 2017.