UNIVERSIDADE FEDERAL DO PARANÁ - SETOR LITORAL

DIEGO LIMA DA COSTA

DESENVOLVIMENTO E IMPLANTAÇÃO DE UM SISTEMA WEB PARA CONTROLE DE AMOSTRAS E OPERAÇÕES DE VAGÕES NO TERMINAL AQUAVIÁRIO DA TRANSPETRO EM PARANAGUÁ

Matinhos – PR

2013

DIEGO LIMA DA COSTA

DESENVOLVIMENTO E IMPLANTAÇÃO DE UM SISTEMA WEB PARA CONTROLE DE AMOSTRAS E OPERAÇÕES DE VAGÕES NO TERMINAL AQUAVIÁRIO DA TRANSPETRO EM PARANAGUÁ

Este Trabalho de Conclusão de Curso foi julgado como requisito parcial à obtenção do título de Bacharel no Curso de Informática e Cidadania da Universidade Federal do Paraná – Setor Litoral, com a seguinte Banca Examinadora:

_____

Prof. Neilor Fermino Camargo.
Professor Orientador

_____

Profª. Mirian Lopes.

_____

Profª. Suzana Cini Freitas Nicolodi.

Matinhos – PR

2013

ATA DE AVALIAÇÃO DA DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos vinte e três dias do mês de setembro de dois mil e treze, às nove horas, no Setor Litoral da Universidade Federal do Paraná, reuniu-se a banca avaliadora do trabalho de conclusão de curso, constituída pela professora Suzana Cini Freitas Nicolodi e pela professora Mirian Cristina Lopes sob a presidência do Orientador, Professor Neilor Fermino Camargo. O Trabalho de Conclusão do Curso de Bacharelado em Informática e Cidadania, do aluno Diego Lima da Costa, sob o título: *"Desenvolvimento e Implantação de um Sistema de Web para Controle de Amostras e Operações de Vagões no Terminal Aquaviário da Transpetro em Paranaguá/PR"*, obteve o conceito APL. O aluno deverá efetuar as correções solicitadas pela banca e entregar a versão final em formato digital via CD-ROOM, até o dia 30 de setembro de dois mil e treze, na assessoria a Câmara do curso de Informática e Cidadania.

Neilor Fermino Camargo
Professor Orientador

Professora Suzana Cini Freitas Nicolodi
Membro da banca avaliadora

Suzana Cini Freitas Nicolodi
Docente
Matrícula 202246
Setor Litoral – UFPR

Professora Mirian Cristina Lopes
Membro da banca avaliadora

Diego Lima da Costa

AGRADECIMENTOS

A minha amada companheira Ana Carla Naumann pela paciência e carinho nos momentos difíceis e contribuições e críticas nos momentos necessários. Aos meus pais e irmãos pelo apoio incentivo e ajuda. Aos meus colegas e amigos principalmente nos momentos difíceis quando a união prevaleceu perante as disputas. Ao meu orientador Prof. Neilor Fermino Camargo pelos conselhos, direcionamentos e orientações. Aos companheiros de trabalho do Terminal Aquaviário da Transpetro de Paranaguá pela oportunidade e confiança depositada.

“A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original.”

Albert Einstein

RESUMO

O objetivo deste trabalho é descrever o desenvolvimento de um sistema Web utilizando o *framework* Ruby on Rails que tem o intuito de auxiliar nos processos de controle de amostras e operações de vagões. Será discutido seus benefícios em relação a forma de gestão utilizada anteriormente pela empresa. Devido à aproximação com a problemática, o empirismo serviu de ferramenta no fomento de ideias, facilitando o processo de construção do sistema. Também serão apresentadas as suas plataformas e ferramentas, dando preferência nas escolhas para tecnologias de *software open source*, salientando a necessidade de uma política organizacional que proporcione um maior entendimento sobre a informatização e suas práticas, sinalizadas pelo desenvolvimento da tecnologia e da inovação que facilita e agiliza o processo trazendo fluidez ao desenvolvimento do trabalho, além de aumentar o controle da informação e melhorar as rotinas dos trabalhadores envolvidos, resultando em maior eficiência, contribuindo para a agilidade nos processos, exaurindo menos os trabalhadores, física e psicologicamente, antes envolvidos com tarefas manuais e repetitivas que geravam stress e não garantiam resultados confiáveis ou de fácil conferência e posterior consulta.

Palavras-chaves: *WEB*, Ruby on Rails, desenvolvimento de sistema, *open source*.

ABSTRACT

The objective of this paper is to describe the development of a Web system using the Ruby on Rails framework that aims to assist in the processes of control samples and operations wagons. Its benefits will be discussed in relation to management previously used by the company. Due to proximity to the problem, empiricism served as a tool in the fomentation of ideas, facilitating the process of building the system. Also it will be presented their platforms and tools used, giving preference of choice to open source software technologies, highlighting the need for an organizational policy that provides a greater understanding of computerization and their practices, marked by the development of technology and innovation that facilitates and streamlines the process bringing fluidity to the development of the work, as well as increasing the information control and improve the involved employees routine, resulting in greater efficiency, contributing to process agility, less employee exhaustion, physically and psychologically, that were previously involved with manual and repetitive tasks that generated stress and could not guarantee reliable or easily conferred results and subsequent consultation.

Keywords: Web, Ruby on Rails, system development, open source.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

Array – Arrays em Ruby são coleções de qualquer objeto, indexadas por inteiros

Back-End – De forma simplificada, pode ser entendido como a parte do código que fica no servidor responsável por atender as solicitações para dados contidos no banco de dados.

Bunker – Termo comumente utilizado para designar combustível para navios.

Byte - Os computadores processam impulsos elétricos, positivos ou negativos, que são representados por 1 ou 0. A cada impulso elétrico damos o nome de bit (*Binary digi*t). Um conjunto de 8 bits reunidos como uma única unidade forma um byte.

Bytecode - Código em bytes.

C – Linguagem de programação compilada de propósito geral.

Free Software Foundation - Organização sem fins lucrativos, fundada em 04 de Outubro de 1985 por Richard Stallman e que se dedica a eliminação de restrições sobre a cópia, redistribuição, estudo e modificação de programas de computadores.

Front-End – Em um sistema WEB são as partes de códigos formadas por HTML, CSS e JavaScript. Parte responsável pela aparência e grande parte da usabilidade do sistema.

GEM – Gema é um pacote ou uma aplicação escrita em linguagem Ruby.

GLP – Gás Liquefeito de Petróleo.

GPL – *General Public License* (Licença Pública Geral).

Hardware - Parte física de um computador.

HTML - *HyperText Markup Language*, Linguagem de Marcação de Hipertexto, é uma linguagem de marcação utilizada para produzir páginas na *Web*.

IDE - *Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento (programa que reúne características e ferramentas de apoio ao desenvolvimento de software).

Intranet - Rede de computadores privada baseada em protocolos da Internet, porém, de uso exclusivo de um determinado local, como, por exemplo, a rede de uma empresa, que só pode ser acessada por seus usuários ou colaboradores internos, tanto internamente como externamente mediante prévia liberação de acesso.

IP – *Internet Protocol* (Protocolo de Internet).

Java - Linguagem de programação orientada a objeto, a linguagem Java é compilada para um *bytecode* que é executado por uma máquina virtual.

JavaScript - linguagem de programação, principal linguagem interpretada por navegadores *WEB*.

Jit - Compilador *just-in-time*, um tradutor que converte, em tempo de execução.

YARV - *Yet Another Ruby Virtual machine*, é uma máquina virtual para Ruby.

Linux - Sistema operacional que utiliza núcleo Linux.

Mac OS X - Sistema operacional proprietário, fabricado e vendido pela empresa americana Apple Inc.

Modais – Modalidades.

Netbeans - IDE gratuito e de código aberto para desenvolvedores de software nas linguagens Java, C, PHP, Ruby, entre outras, e pode ser executado em várias plataformas.

NoSQL – É uma categoria de Banco de dados do tipo não-relacional, estes não utilizam esquemas de tabela fixa.

Open Source - Código aberto, refere-se a software livre. Genericamente trata-se de *software* que respeita as quatro liberdades definidas pela *Free Software Foundation*.

Pascal - Linguagem de programação estruturada.

PDF – *Portable Document Format* (Formato de Documento Portátil).

PHP – *Personal Home Page Tools* (Linguagem de Programação).

Python - Linguagem de programação, interpretada, orientada a objetos, de tipagem dinâmica, *open source*.

Rails ou Ruby on Rails – É um *Framework*, projeto de código aberto escrito na linguagem de programação Ruby.

Ruby – Linguagem de programação, interpretada, orientada a objetos, de tipagem dinâmica, *open source*.

RubyGems - Gerenciador de pacotes para a linguagem de programação Ruby.

SGBD - Sistema de Gerenciamento de Banco de Dados.

Software - Sequência de instruções a serem seguidas e/ou executadas, na manipulação, redirecionamento ou modificação de um dado/informação ou acontecimento. Software também é o nome dado ao comportamento exibido por essa sequência de instruções quando executada em um computador ou máquina semelhante.

SQL – *Structured Query Language* (Linguagem de Consulta Estruturada).

TI – Tecnologia da Informação.

Triggers – Gatilho. É um recurso de programação executado sempre que o evento associado ocorrer.

Tuplas - Similar a uma lista.

URL – *Universal Resource Locator*. Localizador Uniforme de Recursos.

UML - *Unified Modeling Language*. Linguagem de modelagem.

Views - Visão ou vista de um agrupamento de informações.

WEB - *World Wide Web*. Rede de alcance mundial.

XML – *Extensible Markup Language*. Linguagem de Marcação Estendida.

SUMÁRIO

INTRODUÇÃO

Este é um trabalho de conclusão de curso para obtenção do título de bacharel em Informática e Cidadania pela Universidade Federal do Paraná setor Litoral e em seu decorrer será descrito o processo de desenvolvimento de um sistema *WEB* utilizando o *framework* Ruby on Rails. Este sistema *WEB* é utilizado como ferramenta no controle de amostras e operações de vagões em um terminal aquaviário.

Por sua vez este sistema tem objetivos específicos, tais como facilitar o armazenamento de dados, agilizar o acesso a informação, entre outros.

Com os avanços na área de gestão e suas ferramentas cada vez mais informatizadas cria-se um caminho natural para meios digitais no controle das informações, sendo esta uma forma de dinamizar os processos de trabalho, organizando-os e criando banco de dados a serem utilizados como informações para possíveis tomadas de decisões.

Neste sentido, o sistema desenvolvido pode ser visto como dois módulos na tentativa de amenizar dois problemas distintos, o primeiro tem relação com o controle operacional das atividades com vagões, já o segundo tem seu domínio nas atividades de identificação de amostras. Como proposta de solução o projeto teve sua origem na vivencia acadêmica e convívio com os colaboradores do Terminal da Transpetro. Vivenciando este meio, surgiu a ideia de contribuir para a uma melhor forma de trabalho e para a aplicação de conhecimentos adquiridos. Devido à aproximação com a problemática, o empirismo serviu de ferramenta no fomento de ideias, facilitando o processo de construção do sistema.

A busca de ferramentas que possibilitem a maior organização dentro das empresas traz a necessidade de desenvolver novas tecnologias que possam facilitar tais ações. Dentro deste contexto, o sistema desenvolvido utilizou recursos de banco de dados SQL e ferramentas prontas, tais como o sistema operacional Linux, o sistema de virtualização Virtual Box, o SQLite, DIA Diagram, Aptana Studio e o *framework* Ruby On Rails.

Serão apresentados os diagramas de linguagem gráfica UML, os atores e casos de uso básicos que geraram o início do desenvolvimento do sistema e o diagrama de classes que deu origem aos primeiros códigos que deram sentido para as primeiras utilidades ao funcionamento do sistema. Permeando algumas das particularidades do *framework* citado.

Por fim, foi considerado que os objetivos foram alcançados, porém como os processos de uma organização são dinâmicos, suas necessidades também mudam e geram novas demandas.

1 OBJETIVO GERAL

O objetivo deste trabalho é descrever o desenvolvimento de um sistema Web que tem o intuito de auxiliar nos processos de controle de amostras e operações de vagões no terminal aquaviário da Transpetro, empresa subsidiária da Petrobras, situada na área portuária da cidade de Paranaguá no estado do Paraná.

Através deste trabalho, advindo do desenvolvimento do sistema, pretende-se disseminar e incentivar práticas de criação e utilização de sistemas informatizados integrados com o uso e administração de sistemas operacionais, banco de dados, linguagens e técnicas de programação.

Para isto, será necessário abordar as tecnologias utilizadas para o desenvolvimento da solução.

1.1 OBJETIVOS ESPECÍFICOS DO SISTEMA.

- Facilitar o armazenamento de dados;
- Gerar cruzamento de dados;
- Facilitar o gerenciamento do processo de trabalho;
- Agilizar o acesso a informação;
- Reduzir erros humanos;
- Reduzir o tempo gasto com burocracia;

2 JUSTIFICATIVA

Tanto o sistema quanto este trabalho, que o descreve, são tentativas de contribuir para o desenvolvimento tecno-científico da região, de forma prática e teórica, através da divulgação deste trabalho no meio acadêmico e seu uso em outras aplicações.

Este trabalho também fornecerá suporte no processo de gestão da tecnologia da informação desenvolvida.

E como em todo sistema, para dar continuidade ao seu desenvolvimento e criação de novas funcionalidades, o processo de documentação se faz necessário, visando este aspecto o trabalho também ajudará no processo de gestão e aprimoramento da Tecnologia da Informação criada para atender as necessidades dos clientes.

Sendo importante empreender novas propostas e ações que possam ajudar no desenvolvimento tecnológico do litoral paranaense e também de outras realidades sociais, o trabalho em questão também está focado neste sentido.

3 PROBLEMATIZAÇÃO

A primeira questão levantada foi em relação ao controle operacional das atividades com vagões.

Havia a necessidade de averiguar as quantidades de *bunker* movimentadas no terminal, isto era feito através de um processo muito suscetível à erros humanos, pois esta conferência era quase que toda feita manualmente através de formulários impressos. Estes formulários eram recebidos da empresa responsável pelo transporte ferroviário e quando chegavam as composições de vagões os formulários eram utilizados para verificar o peso de cada vagão e após a conferência de todos os vagões esses dados eram passados para uma planilha eletrônica. Ao final do dia essa planilha era utilizada para quantificar o total de *bunker* recebido através dos vagões.

Como continuar a realizar esta sistemática de trabalho e também reduzir erros gerados como por exemplo a conferência de um vagão que não estava sendo descarregado, ou a necessidade de verificar uma quantidade grande de formulários para encontrar um determinado vagão, pois a ordem das notas dificilmente correspondia a ordem de chegada das composições?

Já a segunda questão tinha seu domínio nas atividades de identificação de amostras. Dentro da atividade operacional que abrange o recebimento e/ou envio de derivados de petróleo, há a necessidade de recolher amostras de produto para análise laboratorial para verificar suas propriedades físico-químicas e assegurar que estejam dentro de padrões pré determinados.

Para esta atividade era utilizada a etiquetagem manuscrita destas amostras.

O processo consistia em preencher manualmente etiquetas de papel com dados (procedência, data, hora, produto, responsável, etc...) para cada amostra. Para exemplificar, em uma operação com navio, são geralmente três exemplares para cada amostra e uma operação rotineira pode gerar cerca de até 24 etiquetas, que poderia consumir até 40 minutos de preenchimentos repetitivos. Este processo ainda se tornava mais complicado na questão de controle destas amostras que era feito a partir de uma sequência numérica de recebimento, já que era necessário registrar cada amostra em um livro guardado em um setor muitas vezes diferente de onde eram feitas as etiquetas, necessitando que um empregado estivesse no local para verificar e informar a sequência numérica a ser utilizada.

O que poderia ser feito para manter o bom andamento do processo e diminuir não conformidades no preenchimento, gerando menor retrabalho, atraso, conflitos burocráticos e estresse para as equipes envolvidas?

4 HISTÓRICO

A proposta pedagógica da UFPR Litoral é baseada em projetos e desenvolvida junto às comunidades locais e busca contribuir e incentivar o desenvolvimento científico, econômico, social, ambiental e cultural.

O Projeto Político Pedagógico (PPP) da UFPR Litoral tem por objetivo a promoção da educação pública integrada, para assim desenvolver de forma sustentável toda a região litorânea do Paraná.

Esta proposta organiza o currículo de forma diferenciada, pois ao invés de disciplinas, os estudantes passam por módulos que são mais flexíveis, instigantes e abertos as demandas de cada turma.

São três os grandes eixos de aprendizagem. Fundamentos Teóricos Práticos (FTP), Projetos de Aprendizagem (PA's) e Interações Culturais e Humanísticas (ICH).

O currículo fundamentado na educação por projetos estimula o estudante a construir os saberes, integrando diversas áreas do conhecimento.

Visando os preceitos deste meio, surgiu a ideia de contribuir para a uma melhor forma de trabalho e a aplicação dos conhecimentos gerados a partir do ambiente vivido, dando origem ao presente trabalho.

Desta forma para expor a abrangência deste meio, se faz necessário um breve relato sobre as organizações que o incentivaram.

4.1 UFPR LITORAL

A criação da UFPR Litoral teve seu ano de início em 2004, sendo as atividades iniciadas somente no segundo semestre de 2005.

É em 2007 que o Campus tornou-se Setor. Já no seu início o projeto surge com um diferencial na educação pública pois muda o modelo vigente de transmissão do conhecimento, pelo qual os estudantes têm as aulas expositivas como principal fonte de formação.

Seu objetivo é contribuir para a formação em todos os aspectos dos estudantes. Com isso, a Universidade deixa de ser apenas um caminho para a inserção no mercado de trabalho e passa a ser um espaço em que se é possível desenvolver uma trajetória de vida, estimulando

habilidades como a interação com a comunidade, o empreendedorismo e o poder de envolvimento com as pessoas

4.2 TRANSPETRO

Há quinze anos atuando no setor de logística da Petrobras, a Transpetro (Petrobras Transporte S.A.) tem sua história marcada por inúmeras conquistas no campo estratégico onde atua. Presente em vários estados, a companhia é responsável pela operação de mais de 14 mil quilômetros de oleodutos e gasodutos e também por 48 terminais e 60 navios em sua frota.

Alcançando marcas expressivas no ramo, em 1998 quando foi criada já tinha sob sua responsabilidade a operação de terminais estratégicos para o país, hoje há em seu portfólio 20 terminais terrestres e 28 aquaviários que garantem o abastecimento de petróleo e seus derivados por todo país. O programa de modernização e expansão da frota impulsiona a indústria naval brasileira que ressurge com grande força atrelada aos investimentos ao seu setor de atuação.

Buscando maiores desafios a Transpetro investe em novas modalidades ampliando seu portfólio, é o caso do transporte de etanol por vias fluviais, esse programa almeja a criação de 20 comboios tendo cada um, um empurrador e quatro barcaças previstas para serem entregues ainda este ano. O transporte hidroviário traz vantagens ambientais com a menor emissão de poluentes e impactos ao meio ambiente.

O terminal aquaviário de Paranaguá que foi inaugurado em 1997, tem a área arrendada pela Associação dos Portos de Paranaguá e Antonina (APPA) e trabalha interligado com a refinaria Presidente Getúlio Vargas por modais rodoviários e ferroviários, escoando derivados e operando o poliduto bidirecional Olapa (Oleoduto Araucária – Paranaguá) além de fornecer *bunker* para navios no porto de Paranaguá.

Com o início da exploração do pré-sal, a Transpetro, vem se preparando para atender a essa nova necessidade que surge, antecipando soluções de logística e ampliando alguns dos seus terminais.

5 METODOLOGIA

A metodologia utilizada neste trabalho foi de pesquisa bibliográfica para criação de sistemas informatizados de gerenciamento e armazenamento de dados, durante este processo verificou-se a necessidade de buscar informações na Web, para auxiliar a construção do sistema, sendo utilizadas pesquisas na internet de artigos e trabalhos relacionados ao tema.

Devido à aproximação com a problemática, o empirismo serviu de ferramenta no fomento de ideias, facilitando o processo de construção do sistema, que para Yin,

> "o método da pesquisa empírica, que investiga um fenômeno contemporâneo, dentro de um contexto real de vida no qual as fronteiras entre fenômeno e contexto são claramente evidentes e no qual, múltiplas fontes de evidências são usadas." (Yin, 1989, p. 23)

Ainda buscou-se identificar as relações entre TI e suas múltiplas funções em sua aplicabilidade como, cruzamento de dados, agilidade no processo de trabalho e produção de informação.

6 RECURSOS DE TI

A busca de ferramentas que possibilitem a maior organização dentro das empresas traz a necessidade de desenvolver novas tecnologias que possam facilitar tais ações. Dentro deste contexto a área de tecnologia da informação tem mostrado empenho em aumentar a gama de ferramentas disponíveis no mercado. Verifica-se que informação é a palavra chave e deve englobar um ciclo de construção e distribuição, para que esta seja compreendida e aproveitada da melhor forma.

O conceito de inovação expressado a seguir, contextualiza a necessidade de tais recursos.

Para REIS "A Inovação Tecnológica poderá se definir como uma nova ideia, um acontecimento técnico incontínuo que, após determinado período de tempo é desenvolvido até se tornar prático e então, usado com sucesso." (REIS, 2007, pg. 39)

Desta forma pode-se compreender o processo de construção da inovação tecnológica sendo obtida em fases, tendo em seu primeiro momento o domínio da tecnologia empregada para a sua operacionalização, em seguida ocorrem as adaptações que permeiam os procedimentos e tendem a aperfeiçoar tais inovações.

No entanto o êxito de um novo produto ou processo não está atrelado ao valor da sua descoberta ou na sua criatividade, é necessário que este seja disponibilizado e principalmente utilizado, portanto deve trazer possibilidades de novas oportunidades, para então ser considerado uma inovação.

Desta forma compreende-se um sistema de informação como sendo a congregação de pessoas, tecnologias da informação, estruturas da organização, com procedimentos e processos interligados de forma a proporcionar à organização as informações que ela necessita. Neste auxílio, a computação tem evoluído a desenvolver sistemas cada vez mais complexos, trabalhando em redes amplas tendo sempre como objetivo mostrar uma informação coesa, correta, para que seja útil a quem possa utilizar.

6.1 BANCO DE DADOS

Pode-se entender por banco de dados qualquer sistema que reúna e mantenha organizada uma série de informações relacionadas a um determinado assunto em uma determinada ordem.

Para Ricarte define-se banco de dados relacional da seguinte forma:

> "Um banco de dados organiza seus dados em relações. Cada relação pode ser vista como uma tabela, onde cada coluna corresponde a atributos da relação e as linhas correspondem as tuplas ou elementos da relação. Em uma nomenclatura mais próxima àquela de sistemas de arquivos, muitas vezes, as tuplas são denominadas registros e os atributos, campos. Um conceito importante em um banco de dados relacional é o conceito de atributo chave, que permite identificar e diferenciar uma tupla de outra. Por meio do uso de chaves é possível acelerar o acesso a elementos (usando índices) e estabelecer relacionamentos entre as múltiplas tabelas de um sistema de banco de dados relacional." (Ricarte, 2007)

Este processo inclui um forte embasamento matemático, apoiando os conceitos utilizados nos bancos de dados relacionais trazendo uniformidade na manipulação dos sistemas utilizando a linguagem *Structured Query Language* (SQL).

O banco de dados sempre fornecerá aspectos reais, sendo um referencial para se extrair uma ampla estrutura de informações com múltiplas possibilidades por meio de cruzamentos de chaves. É necessário ter uma interface entre o usuário e o banco de dados, tal interação se dá por meio especifico de sistemas que acessam a gama de informações geralmente por linguagem SQL.

Essas ferramentas possibilitam dinamizar a criação e a utilização de seus dados definindo uma maior estruturação e funcionalidade, tornando os dados mais reais e autênticos, permitindo que o processo de utilização da informação seja feita com maior agilidade.

6.2 FRAMEWORK

Um f*ramework* provê uma solução para uma família de problemas semelhantes, usando um conjunto de classes e interfaces que mostra como decompor a família de

problemas, este conjunto de classes deve ser flexível e extensível, suficiente para permitir o desenvolvimento de aplicações com menor esforço, sendo assim necessário apenas a especificação das particularidades de cada aplicação.

O *framework* deve ser uma aplicação pronta, quase completa, mas com partes específicas faltantes, estas partes consistem no trabalho que o desenvolvedor deve realmente se preocupar em resolver, para que o sistema no final, atenda a uma ou várias necessidades específicas. Resumidamente, um framework deve ter características como, robustez, reusabilidade, extensibilidade e eficiência.

6.3 FERRAMENTAS UTILIZADAS

Para alcançar o objetivo de criação do sistema web é fundamental o entendimento e familiarização de uma gama de ferramentas que o possibilitem. Por este motivo esta seção é dedicada a descrever as ferramentas e tecnologias principais.

6.3.1 Ubuntu

Sistema operacional baseado em Linux criado pela comunidade (usuários e entusiastas) utilizado em *laptops*, *desktops* e servidores. Este sistema é oferecido gratuitamente, sendo lançada uma nova versão a cada seis meses para servidores de *desktops*, sempre visando a segurança, tendo disponíveis atualizações por até 18 meses.

A versão de longo tempo de suporte (LTS) oferece três anos de suporte para *desktops* e cinco anos para servidores. Esta versão também é disponibilizada gratuitamente, assim como todas as outras. Permitindo ter um sistema funcional e ágil o instalador gráfico realiza a instalação padrão em até 30 minutos.

Assim que instalado, o sistema está imediatamente pronto para o uso, a versão para *desktop* oferece um conjunto completo de aplicativos para produtividade, internet, imagens, jogos, entre outras ferramentas.

A escolha do sistema operacional Ubuntu para a utilização como base de estrutura para suportar a aplicação está ligada ao conhecimento, maturidade e familiarização tida com o

mesmo e pelo fato de ele suportar sem grandes problemas a estrutura da aplicação em ambiente de produção e teste.

6.3.2 Virtual Box

Este programa de virtualização da Oracle (empresa multinacional de tecnologia e informática dos Estados Unidos) permite instalar e executar diferentes sistemas operacionais em um único computador. Sendo possível executar o Linux dentro do Windows, o Windows dentro do Mac, o Mac dentro do Windows ou trabalhar com todos os sistemas na mesma máquina, tendo como limitante para estas combinações, o hardware do computador.

Fez-se necessário a utilização de um ambiente de virtualização, pois no servidor disponibilizado para manter o sistema em questão já rodava uma série de aplicações e seu sistema operacional nativo era Windows, dificultando a migração para o ambiente de produção. Houve certo receio no começo, pois não existia a certeza de que o ambiente iria funcionar de acordo com o esperado, rodando dentro de um ambiente de virtualização com capacidade de máquina restrita, porém isto não ocorreu e o desempenho atendeu as necessidades.

6.3.3 SQLite

SQLite é um mecanismo para banco de dados SQL embutido. Diferentemente de outros bancos de dados SQL, o SQLite não possui um servidor separado, este lê e escreve diretamente para os arquivos do disco. Ele provê um completo banco de dados. Este programa contém várias tabelas, índices, triggers e views contidos em um arquivo de disco único.

Como SGBD (Sistema de Gerenciamento de Banco de Dados) optou-se pelo SQLite por comodidade, pois ele já vem nativamente com o ambiente de instalação do Ruby on Rails e por atender aos requisitos de desempenho esperado, não foi necessário a adoção de outro SGBD.

6.3.4 DIA *Diagram*

Liberado sobre a licença GPL, Dia é um programa de criação de diagramas, ele é mais voltado para diagramas de uso casual.

Pode ser utilizado para desenhar vários tipos diferentes de diagramas, atualmente conta com objetos especiais que auxiliam em desenhos de diagramas de entidade racional, diagramas UML, fluxogramas de rede e etc...

Para a modelagem dos diagramas foi escolhido o Dia, pois ele possibilita a exportação direta do diagrama de classe para a linguagem Pascal (posteriormente isto será melhor abordado), isto agiliza o processo de geração do código de criação da aplicação para o Rails, reduzindo erros de digitação e otimizando o processo inicial.

6.3.5 Aptana Studio

O *software open source*, Aptana é um IDE gratuito, desenvolvido em Java que suporta PHP, Python, Ruby, Ruby on Rails, HTML, JavaScript, XML e texto comum. É oferecido em multi-plataforma podendo ser rodado em Windows, Linux e Mac OS X.

O Aptana não apresentou, assim como outros IDE's, grandes diferenças para a usabilidade com o Rails. Talvez por falta de conhecimento na configuração do ambiente não foi possível alcançar facilidades como ajuda na criação de código, por exemplo, sugestões de métodos de uma classe e sua documentação, conhecidos como assistente para auto completação de código para o ambiente do projeto em Rails. Mas assim como outras IDE's ele traz este recurso para a linguagem Ruby, e por ter sido o último a ser testado ele acabou sendo o escolhido.

Obs.: após toda a conclusão do projeto, descobriu-se que o Netbeans (da Oracle Corporation) pode ser configurado para os recursos citados, porém não houve tempo hábil para os testes de configurações necessários.

6.3.6 Mozilla Firefox

Este navegador WEB livre, multi-plataforma desenvolvido pela Mozilla Foundation e com a ajuda de centenas de colaboradores foi a escolha mais simples de ser feita, devido a empresa objeto de implantação do sistema utilizá-lo como navegador padrão dentro de suas unidades.

6.3.7 O *Framework* web Ruby on Rails

Ruby on Rails é um *framework* de desenvolvimento web que otimiza a produtividade e permite a criação de código fonte de forma elegante, o que favorece a convenção ao invés da configuração, adotando padrões pré estabelecidos e de boas práticas.

A escolha central do projeto foi a linguagem de programação e seus recursos a serem utilizados, houve algumas tentativas com outras linguagens antes da adoção do Ruby e o *Framework* Rails, a que teve maior aproveitamento, porém não alcançou o desejado foi Java. Apesar de existirem vários recursos e uma grande maturidade da linguagem, na hora de desenvolver e implantar o projeto WEB java com seus *Framework's*, existiu grande dificuldade pois a curva de aprendizado em Java foi muito mais demorada se comparada com Rails.

Em java uma aplicação simples requer um grau elevado de conhecimento, já em Rails isto se tornou mais natural e gratificante, pois era possível ver resultados rápidos de recursos que já poderiam ser utilizados.

Uma imagem de comparação encontrada em alguns sites que fazem uma brincadeira um tanto séria comparando as tecnologias, mostra as literaturas básicas para o desenvolvimento com ambas, onde as pilhas de livros fazem alusão com cada uma delas.

Ilustração 1: Comparação entre literaturas Java e Rails.

Apesar de haver algumas comparações sem muito fundamento, o que foi percebido durante a elaboração do projeto é a simplicidade e produtividade ao utilizar Rails, talvez por ter passado por outras tentativas com tecnologias diferentes e isto ter trazido uma bagagem teórica e um preparo maior quando chegaram aos testes com Rails, ou porque simplesmente Rails é mais simples e mais fácil de desenvolver.

Houveram outras tentativas menores com PHP e Python, porém não ganharam destaque, pois o projeto em Rails já estava bem adiantado e atendendo a expectativa.

Antes de falar mais especificamente sobre o *framework* Ruby on Rails, é necessário falar um pouco sobre a linguagem de programação utilizada por ele, Ruby, como o próprio nome já sugere.

6.3.7.1 A linguagem Ruby

De acordo com o site oficial da linguagem, Ruby é uma linguagem dinâmica, *open source* com foco na simplicidade e na produtividade. Tem uma sintaxe elegante, de leitura natural e fácil escrita.

6.3.7.1.1 A história do Ruby e suas características

Apresentada ao público pela primeira vez em 1995, pelo seu criador, Yukihiro Matsumoto, é uma linguagem orientada a objetos, com tipagem forte e dinâmica, caracterizada pela expressividade que possui, objetivando desde o início que sua linguagem fosse muito simples de ser entendida, para facilitar o desenvolvimento e manutenção de sistemas escritos com ela.

6.3.7.1.2 Instalação do interpretador

Antes do Ruby se tornar popular, havia apenas um interpretador disponível, escrito pelo próprio Matsumoto, em C. Sendo um simples interpretador, sem nenhum gerenciamento de memória muito complexo, nem características modernas de interpretadores, como a compilação em tempo de execução conhecida como JIT. Atualmente a versão mais difundida é a 1.9, conhecida como YARV *Yet Another Ruby VM*, que se baseia em máquinas com recursos mais avançados.

A maior parte das distribuições Linux possui pacotes de uma das últimas versões estáveis, pronto para ser instalado. Seguindo como exemplo no Ubuntu a instalação é:

```
sudo apt-get install ruby
```

Para a elaboração deste projeto foi utilizada a versão 1.9.2 do Ruby.

```
ruby 1.9.2p320 (2012-04-20 revision 35421) [x86_64-linux]
```

Os detalhes de sintaxe e funcionamento da linguagem podem ser encontrados na sua documentação na *WEB*, este assunto será abordado de forma superficial neste trabalho pois não é seu foco principal.

Ruby possui um sistema gerenciador de pacotes avançado, flexível e eficiente, o RubyGems. As gems podem ser compreendidas como pacotes reutilizáveis de código Ruby. No entanto para instalar e utilizar as centenas de gems disponíveis, faz-se necessário instalar além do interpretador Ruby, o RubyGems. Como exemplo simples, segue o comando para instalação no Ubuntu:

```
sudo apt-get install rubygems
```

Neste projeto foi utilizada a versão 1.8.24.

O próprio *Framework* Rails é uma gem, na verdade ele é um conjunto de gem's, e por este motivo é comum encontrar em fóruns de desenvolvedores declarações de que Rails não é um *framework*, mas sim um *meta-framework* (ou seja, um *framework* de *framework's*).

6.3.7.2 Ruby On Rails

Como pilares, o Rails sustenta dois conceitos, *Don't Repeat Yourself* (DRY) não repita a si mesmo, e *Convention over Configuration* (CoC) convenção ao invés de configuração.

O primeiro conceito incentiva o uso da reutilização de código, sendo uma das principais vantagens da orientação a objetos, podendo aproveitar as características de Orientado a Objeto do Ruby, o próprio *framework* encoraja a adotar padrões de projeto mais adequados, para essa finalidade.

O segundo conceito, facilita escrever menos códigos para implementar uma determinada funcionalidade, respeitando alguns padrões de nome e localização de arquivos, nome de classes e métodos.

Em geral as aplicações apresentam um código de maior simplicidade e enxuto por conta desse conceito.

As aplicações que usam o Ruby on Rails são inúmeras e entre elas estão o Twitter, YellowPages, Groupon, Typo (blog open source) e o Spokeo ferramenta de agrupamento de sites de relacionamentos.

O MVC (*Model, View, Controller*) é a arquitetura principal do Rails, baseada no conceito de separar tudo em três camadas, Modelo, Visão e Controlador. É um padrão arquitetural, tendo seus modelos, suas lógicas e suas visualizações bem definidos, trazendo maior simplicidade em se fazer um reparo ou uma mudança, uma vez que essas três partes se comunicam de maneira desacoplada.

Para a instalação do Rails no Ubuntu basta seguir o comando:

```
gem install rails
```

A versão para este projeto é a, Rails 3.1.0.

Mais adiante, quando será discutido o processo de desenvolvimento do sistema, serão abordados novos conceitos do Rails, assim será possível uma relação mais coesa da teoria com a prática, evitando-se conceitos desconexos com suas aplicabilidades.

7 DESENVOLVIMENTO E IMPLEMENTAÇÃO DO SISTEMA

"A atividade de implementação está relacionada com a codificação das classes abstratas e concretas da Estrutura de Aplicação. O resultado desta atividade é o conjunto de classes que implementam o comportamento básico da Estrutura de Aplicação." (CAMARGO, 2003, pg. 20)

Desta forma inicia-se a descrição da elaboração e preparação dos módulos necessários para a criação e execução do sistema.

7.1 MODELAGEM DO SISTEMA

A atividade de modelagem de software consiste em construir modelos para explicar o comportamento ou características do software. Assim para identificar as características e funcionalidades que o software proverá, usam-se técnicas que façam uso dos modelos para sua construção.

> "Um modelo é uma abstração que tem como propósito entender um problema antes de solucioná-lo. A partir de modelos, é possível simular e testar sistemas antes de construí-los, facilitar a comunicação com os usuários e os outros membros da equipe de desenvolvimento, visualizar e reduzir a complexidade dos problemas a tratar." (MAZZOLA, pg. 95)

A linguagem gráfica UML é uma forma comumente usada na modelagem de programas orientados a objeto. Uma linguagem se faz necessária para este processo pois ao se modelar um software, isto normalmente gera a construção de modelos gráficos, usados para simbolizar componentes de software relacionamentos.

Neste sentido a primeira fase do processo foi identificar os casos de uso básicos, que levaram a iniciar o processo de criação do conjunto de classes que iriam representar as entidades que interagem no processo.

7.1.1 Caso de uso

Abaixo está o diagrama de caso de uso que iniciou o processo de criação do sistema.

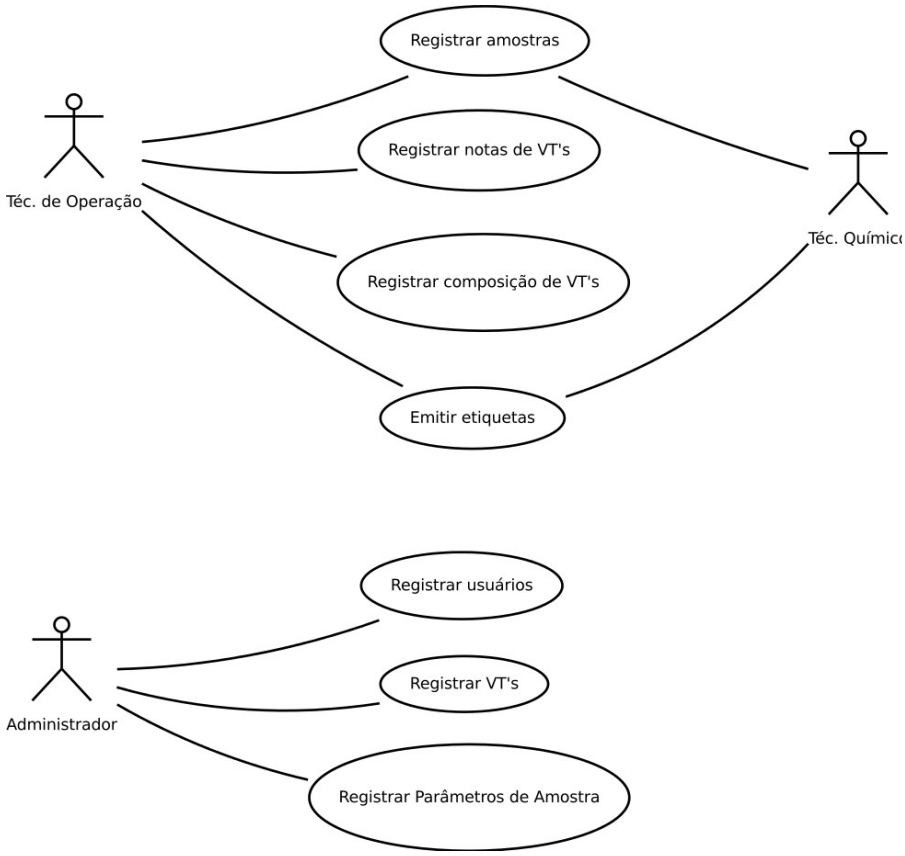


Ilustração 2: Diagrama de caso de uso.

Casos de uso são os processos que fazem parte do domínio da aplicação, eles representam a sequência do início ao fim de um determinado processo.

Atores são entidades que de algum modo participam de um ou mais casos de uso.

Os atores e casos de uso básicos que geraram o início do desenvolvimento do sistema são descritos a seguir.

7.1.1.1 Téc. de Operação

Ator principal de vários processos, ele é o empregado da empresa encarregado de utilizar o sistema a fim de facilitar seu trabalho no dia-dia. Tem os privilégios de registrar as amostras, sendo ele o principal responsável por coletá-las, registrar notas de vagões, registrar composição de vagões e emitir etiquetas.

7.1.1.2 Téc. Quimico

Este ator também é um funcionário da empresa, responsável por elaborar os ensaios necessários para averiguar as características das amostras que chegam até o laboratório. Tem privilégios de registrar amostras (pois eventualmente precisa realizar esta tarefa) e emitir etiquetas.

7.1.1.3 Administrador

Ator funcionário da empresa, com os privilégios de registrar novos usuários, novos vagões e parâmetros de amostras.

7.1.1.4 Registrar amostra

Caso de uso que permite que o usuário cadastre uma ou várias amostras através de um formulário, escolhendo os parâmetros pré-estabelecidos para ela.

7.1.1.5 Registrar notas de VT's

Após receber uma relação de vagões a serem descarregados (nota de VT's), o usuário utiliza este caso de uso para registrar esta nota no banco de dados do sistema.

7.1.1.6 Registrar composição de VT's

Neste caso de uso o usuário registra as composições de vagões, sem se preocupar em procurar em qual nota de vagão estão os dados cadastrados, ele apenas digita os números que identificam os vagões e o sistema irá fornecer o peso, data de carregamento, produto e demais dados já lançados no caso de uso “Registrar notas de VT's”.

7.1.1.7 Emitir etiquetas

Após cadastradas as amostras, o usuário pode necessitar imprimir uma etiqueta para identificá-las no meio físico, pensando nesta necessidade é que este caso de uso foi criado. Basicamente ele recebe como parâmetro algumas amostras e devolve um relatório em PDF para o usuário poder imprimir as etiquetas de identificação.

7.1.1.8 Registrar usuários

Caso de uso de acesso exclusivo ao administrador, necessário para poder criar novos usuários do sistema.

7.1.1.9 Registrar VT's

Este caso de uso serve para que novos vagões sejam cadastrados.

7.1.1.10 Registrar Parâmetros de Amostra

Para que se possam registrar amostras são necessários alguns parâmetros, como origem, produto, operação etc... e estes parâmetros são cadastrados através deste caso de uso.

7.1.2 Classes e suas descrições

Abaixo está o diagrama de classes que deu origem aos primeiros códigos que deram sentido para as primeiras utilidades ao funcionamento do sistema.

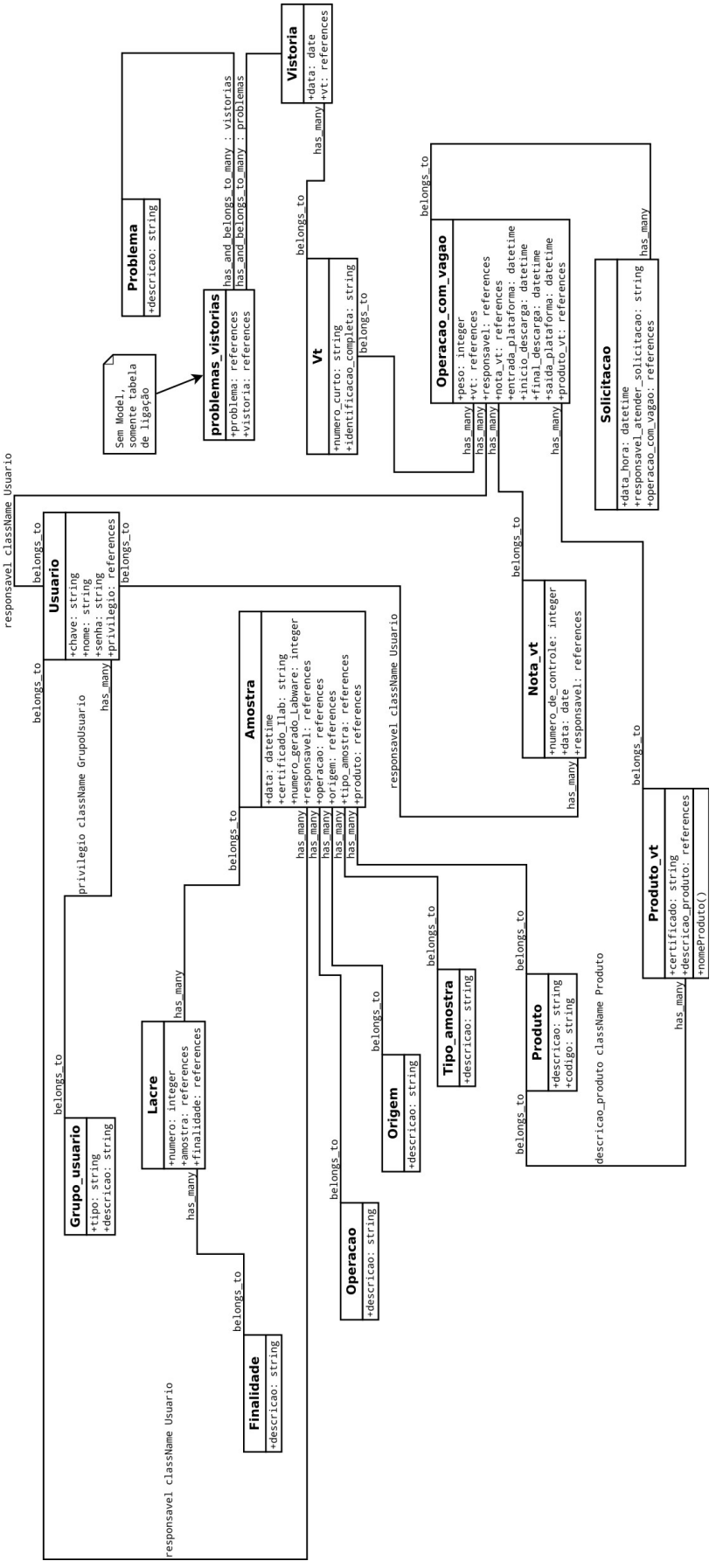


Ilustração 3: Diagrama de classe.

Este diagrama passou por algumas atualizações após ser usado como base para a criação das classes principais, na tentativa de manter parte da documentação atualizada, porém após uma busca por uma *gem* que realiza-se este trabalho de forma automática foi encontrada a *RailRoady* uma *gem* que gera os diagramas UML da aplicação e também uma documentação do código fonte dos objetos do sistema, desta forma não ouve mais sentido em atualizar este diagrama, visto que esta *gem* o faz de maneira mais eficiente.

Os diagramas UML gerados por esta *gem* estão nos anexos e representam o estado atual do sistema.

Para evitar erros, os nomes das classes e seus parâmetros foram criados sem acento ou cedilha.

7.1.3 Do digrama para o esqueleto do sistema

Após ter o diagrama de classe com as principais entidades do sistema, é necessário transforma-lo em algo que possa ser entendido pelo Rails para a criação do MVC (*Model View Controller*) de cada uma delas, já que não haveria sentido digitar tudo novamente.

Para isto o Rails disponibiliza um comando que facilita a criação das três camadas de uma entidade, o *scaffold.*

O *scaffold* ou andaime, otimiza a criação da base para a construção de uma aplicação.

Ele gera toda a parte de MVC relativa ao modelo que se é passado como parâmetro, criando assim toda a estrutura inicial necessária para, as lógicas de listagem, inserção, edição, busca e suas visualizações. É possível também criar a partir dele o código que irá gerar a tabela do banco de dados para este modelo.

Com estes recursos é possível então utilizar o gerador de código disponibilizado pelo programa DIA. E a partir do diagrama de classe exportar para código pascal.

O código pascal foi escolhido pois sua sintaxe é próxima do comando *Scaffold* o que geraria pouco trabalho para adaptação.

Como exemplo, tem-se um trecho do código em pascal gerado pelo DIA logo abaixo.

```
Amostra = class
  public
    data : datetime;
    certificado_Ilab : string;
    numero_gerado_Labware : integer;
    observacao : text;
    responsavel : references;
    operacao : references;
    origem : references;
    tipo_amostra : references;
    produto : references;
  End;
```

Ilustração 4: Trecho do código em pascal gerado pelo DIA.

A partir deste código é possível tratá-lo com o Gedit, ou qualquer outro editor simples de texto que tenha o recurso de "localizar e substituir". E após o tratamento o código *scaffold* pronto para gerar a entidade "Amostra" pode ser visto na sequencia.

```
Rails generate scaffold Amostra data:datetime certificado_Ilab:string
numero_gerado_Labware:integer responsavel:references operacao:references
origem:references tipo_amostra:references produto:references
```

Este "atalho" ou "artifício" apresentado para a entidade "Amostra" foi utilizado para gerar toda a estrutura inicial do sistema, reduzindo drasticamente o tempo para iniciar os ajustes no sistema.

Os códigos completos podem ser visualizados na íntegra nos anexos. Estes códigos podem apresentar diferenças em relação ao código fonte do sistema, visto que ele foi usado para gerar a base do sistema, após isto houveram alterações no código fonte para atender as necessidades das regras de negocio.

O Rails adota co1mo convenção uma estrutura de arquivos em que os modelos, controladores e visualizações são colocados em seus lugares específicos, isto gera, entre outras facilidades, agilidade para manutenção de código.

Esta estrutura obedece uma ordem simples que após a adaptação do desenvolvedor torna o trabalho mais intuitivo. Na sequência é possível visualizar esta estrutura e uma descrição sucinta dos principais diretórios.

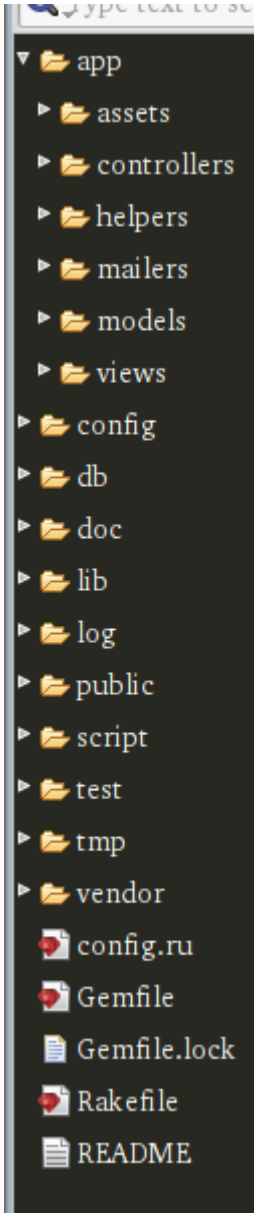


Ilustração 5: Estrutura de diretórios Rails.

app - quase que 100% de toda a aplicação, códigos criados, o MVC dividido em diretórios, ficam neste diretório, não por acaso que o nome do diretório é a abreviatura de aplicação. Em

"*app/controllers*" ficam os controladores, "*app/models*" os modelos e em "*app/views*" as visualizações.

config - As configurações e ajustes da aplicação ficam no *config* (configuração).

db - O que é relativo ao banco de dados fica em *db* (*data base*).

7.1.4 Descrição das classes

As classes geradas pelo "Sacaffold" bem como suas primeiras modificações serão descritas a seguir nos itens que seguem.

7.1.4.1 Amostra

Classe que representa uma amostra, com os seguintes parâmetros:

- data:datetime – data e horar que a amostrar foi retirada.
- certificado_Ilab:string – parâmetro que representa o certificado de qualidade da amostra, identificação gerada por outro sistema interno.
- numero_gerado_Labware:integer - parâmetro que representa a identificação da amostra em outro sistema interno.
- responsavel:references – identificação do usuário que registrou a amostra, este parâmetro é do tipo "references" pois este tipo diz ao Rails na hora de criação desta classe, que ela está associada a uma outra classe, neste caso é a classe usuário, através da associação

```
belongs_to :responsavel, :class_name => 'Usuario'
```

inserida no "model" amostra.

- operacao:references – indica que operação a que amostra está associada, também é "references" e portanto indicará uma ligação com chave estrangeira para a tabela

operação, assim como no caso anterior, porém neste caso a associação tem o mesmo nome da classe, “

```
belongs_to :operacao
```

- origem:references – indicação de onde foi retirada a amostra, ou seja, sua origem, chave estrangeira.
- tipo_amostra:references – qual o tipo de amostra, composta, de nível, etc..., tambe é uma chave estrangeira.
- produto:references – de que produto é constituído a amostra, chave estrangeira.

7.1.4.2 Composicao_de_vagao

A classe Composicao_de_vagao estabelece um conjunto de informações para serem utilizadas por um ou mais vagões que tenham os seguintes atributos em comum:

- entrada_plataforma:datetime – data e hora que a composição entrou na plataforma de vagões.
- inicio_descarga:datetime – data e hora que iniciou a descarga do produto contido nos vagões da composição.
- final_descarga:datetime – data e hora do final de descarga.
- saida_plataforma:datetime – data e hora que o último vagão da composição sair da plataforma.
- responsavel:references – chave estrangeira da tabela “Usuario”, este atributo é preenxido automaticamente pelo sistema, pois assume-se que o responsável é o usuário que está logado e por sua vez preenchendo os campos do formulário.
- fechamento_diario_vagao:references – chave estrangeira da tabela “Fechamento_diario_vagao”.

7.1.4.3 Fechamento_diario_vagao

A classe “Fechamento_diario_vagao” representa um modelo para um agrupamento de várias instâncias da classe “Composicao_de_vagao”. Ou seja, um fechamento diário terá uma ou mais composições de vagões e cada composição pertencerá a um fechamento. Esta classe foi criada para que existisse uma forma simples de se quantificar os dados necessários para efetuar o “corte” lógico entre as várias descargas de composições de vagões.

Assim como em outras classes que possuam uma associação de 1 para N, esta associação no Rails é feita através do relacionamento inserido no modelo, neste caso no modelo “FechamentoDiarioVagao” e o relacionamento é

```
has_many :composicao_de_vagaos
```

Visto que o relacionamento é feito na tabela “composicao_de_vagao” através da chave estrangeira que aponta para a instância da tabela “fechamento_diario_vagao”, o único atributo que esta classe terá na sua tabela será:

- datahora:datetime – data e hora do fechamento, que representa o tempo que separa um fechamento de outro, ou corte lógico nas descargas das composições.

7.1.4.4 Finalidade

Classe que representa qual é a finalidade de um frasco de uma amostra, arquivo, análise, etc...

A amostra é representação de um determinado produto em si, uma finalidade pode ser o destino que cada frasco deste produto terá. Por exemplo, uma amostra de água foi retirada de um determinado local, esta amostra será arquivada e analisada, para tal são necessários dois recipientes um para arquivo e outro para análise, cada recipiente terá uma finalidade.

O número de finalidades de uma amostra também determinará o número de etiquetas que esta amostra terá, já que cada frasco necessita de uma identificação.

- descricao:string – descrição da finalidade.

7.1.4.5 Grupo_usuario

A classe grupo de usuários foi criada com o intuito de restringir os acessos aos recursos do sistema de acordo com cada usuário.

Assim, se um usuário é do tipo visitante ou técnico químico por exemplo, deve acessar recursos somente do seu grupo.

Inicialmente esta forma de resolver o problema de acesso pareceu atender a necessidade que se esperava, porém seria mais fácil e mais reutilizável se cada usuário tivesse como atributos do tipo booleano para cada validação necessária.

- tipo:string – atributo que indica (de forma resumida) qual tipo é o grupo, técnico químico, visitante, etc...
- descricao:string – descrição completa do grupo de usuários.

7.1.4.6 Lacre

Lacre é um dispositiva contra violação, geralmente de plástico e possui uma numeração que o identifica.

Dependendo da finalidade que uma amostra terá, ela pode exigir um lacre, desta maneira todo lacre tem uma finalidade associada.

- numero:integer – numeração que identifica o lacre.
- amostra:references – chave estrangeira que associa o lacre a uma amostra.
- finalidade:references – chave estrangeira que associa o lacre a uma finalidade.

7.1.4.7 Nota_vt

Documento físico entregue para conferência dos vagões.

- numero_de_controle:integer – número que identifica o documento.
- data:date – data de emissão do documento.
- responsavel:references – responsável pela digitação, preenchido automaticamente pelo usuário logado.

7.1.4.8 Operacao

Uma amostra necessariamente tem uma operação associada e uma operação pode ter várias amostras associadas.

- descricao:string – descrição da operação, carga de navio, descarga de vagão, etc...

7.1.4.9 Operacao_com_vagao

Diferente da classe anterior que fazia parte do contesto de amostragem, esta classe representa uma operação de um vagão.

Ou seja, um vagão pode carregar e descarregar várias vezes e cada vez que ele faz um destes ciclos e está realizando uma operação (operação com vagão).

- peso:integer – peso do produto que o vagão contém.
- vt_carregado:boolean – este atributo indicará se o vagão está carregado ou vazio, pode-se ler também como sendo a indicação se a operação de um determinado vagão já acabou ou não.
- vt:references – chave estrangeira que indica qual vagão a operação está associada.

- nota_vt:references – referência que indica qual nota de vagão originou a operação. Uma nota geralmente tem várias operações de vagões. Basicamente uma operação de vagão inicia quando uma nota é lançada e associada a ela.
- produto_vt:references – produto que contido no vagão.
- composicao_de_vagao:references – após a operação ser lançada e associada a uma nota, ela inicia seu ciclo e após sua associação com uma composição e término da descarga desta composição seu ciclo se encerra.

7.1.4.10 Origem

Origem da amostra, descreve de onde veio a amostra.

- descricao:string – descrição da origem.

7.1.4.11 Problema

Problema que um vagão pode ter. Usado para relatar anormalidades para a prestadora de serviço.

- descricao:string – descrição do problema, gotejamento, sujeira, etc...

7.1.4.12 Produto

Classe que descreve qual de produto é constituída uma amostra.

- descricao:string – descrição do produto, água, gasolina, etc...
- codigo:string – código que identifica o produto pelos setores envolvidos.

7.1.4.13 Produto_vt

Classe que descreve qual produto uma operação com vagão está associada.

- certificado:string – identificação do certificado de análise do produto.
- descricao_produto:references – chave estrangeira que indica qual produto está associado ao produto do vagão. Na classe "ProdutoVt" existe uma associação com a classe "Produto",

```
belongs_to :descricao_produto, :class_name => 'Produto'
```

e de forma análoga também existe sua correspondência na classe "Produto",

```
has_many :produto_vts, :foreign_key => :descricao_produto_id
```

7.1.4.14 Solicitacao

Registra a solicitação feita com a empresa prestadora do serviço de transporte rodoviário para a retirada dos vagões.

- data_hora:datetime – data e hora do contato com o representante da empresa prestadora, geralmente via telefone.
- responsavel_atender_solicitacao:string – nome do representante da prestadora.
- composicao_de_vagao:references – a qual composição a solicitação foi feita. Ou seja, qual composição está pronta para ser retirada da plataforma ferroviária.

7.1.4.15 Tipo_amostra

Tipo da amostra, explica de qual tipo é a amostra.

- descricao:string – descrição. De topo, meio, composta, etc...

7.1.4.16 Usuario

Modelo de usuário, utilizador do sistema.

- chave:string – chave interna do usuário, fornecida pela empresa.
- nome:string – nome completo.
- nome_guerra:string – nome como o usuário é conhecido no meio que trabalha.
- observacao:string – atributo usado para indicar se o usuário está em atividade ou se está inativo.
- senha:string – senha do usuário.
- privilegio:references – chave estrangeira para a tabela privilégio.

7.1.4.17 Vistoria

Através da vistoria é que um ou vários problemas serão associados a um vagão.

Uma vistoria pode ter vários problemas e um problema pode constar em várias vistorias, criando uma relação de N para N.

- data:date – data de registro da vistoria.
- vt:references – relação com a tabela de vagões. Um vagão pode ter várias vistorias, mas uma vistoria pertence apenas a um vagão.

7.1.4.18 Vt

Classe que representa um vagão.

- numero_curto:string – identificação curta de uma vagão, os cinco últimos códigos (9929-1).
- identificacao_completa:string – a identificação contendo todos os códigos da identificação de um vagão (TSC-639929-1).

7.1.4.19 problemas_vistorias

Esta é apenas uma tabela, "problemas_vistorias" não é uma classe, esta tabela serve para realizar o relacionamento de N para N entre as entidades Problema e Vistoria. Como padrão, o método que cria o relacionamento muitos para muitos no Rails "has_and_belongs_to_many", possui uma convenção para se nomear a tabela que persistirá os relacionamentos, ela deve estar em ordem alfabética e separada por "underscore" (_). Outra informação que merece destaque é o fato desta tabela não ter a coluna "id", mas apenas duas colunas, "problema_id" e "vistoria_id".

Para que o Rails entenda que existe um relacionamento de N para N entre as duas classes, também é necessário manter as declarações

```
has_and_belongs_to_many :vistorias
```

na classe "Problema" e

```
has_and_belongs_to_many :problemas
```

na classe "Vistoria".

Não serão abordados aqui todos os métodos dos relacionamentos do Rails, mesmo porque não é este o objetivo, para tal a API do f*ramework* e os guias especializados no assunto podem ser consultados. Para a elaboração do presente trabalho foi consultado massivamente o guia (http://guides.rubyonrails.org/association_basics.html).

7.2 PRINCIPAIS MODIFICAÇÕES DO *SCAFFOLD*

Após a criação do esqueleto das entidades na arquitetura do Rails, são necessários ajustes e modificações para que o sistema se comporte de acordo com o esperado e atenda as necessidades do cliente, que neste caso são os usuários do sistema.

Como o objetivo aqui é explanar sobre a forma do uso e as possibilidades da tecnologia

utilizada, não faria sentido abordar todas as modificação realizadas, e todas as particularidades do *framework* e da linguagem empregadas no processo, assim optou-se por descrever algumas das modificações de maior relevância. Porém os códigos do sistema podem ser encontrados nos anexos para possíveis consultas mais detalhadas.

7.2.1 Modificações em “class AmostrasController” (Controlador)

A primeira modificação encontrada na casse “AmostrasController” são os filtros de acesso aos métodos:

```
before_filter :liberarAcesso, :except => [:index, :show, :new]
before_filter :liberarExcluir, :only => [:destroy]
```

Ilustração 6: “AmostrasController” filtros de acesso.

estes dois filtros servem para verificar qual privilégio o usuário tem e de acordo com este privilégio liberar ou não o acesso aos métodos.

```
def liberarAcesso
  if @usuario.privilegio.id == 2 or @usuario.privilegio.id == 3
  return true
  else
    redirect_to "/static/grupo_diferente"
  return false
  end
end

def liberarExcluir
  if @usuario.privilegio.id == 3
  return true
  else
    redirect_to "/static/grupo_diferente"
  return false
  end
end
```

Ilustração 7: “AmostrasController” filtros de acesso métodos.

O método “index” foi modificado para evitar sobrecarga no banco de dados, ele recebe como parâmetro uma variável “:data_inicial” e caso ela esteja vazia ele retornará apenas os últimos cem registros, caso contrário acessa outra rotina que devolve os registros de acordo com um período selecionado pelo usuário.

```
# GET /amostras
# GET /amostras.json
def index
    if params[:data_inicial] == nil
      @amostras = Amostra.last(100)
    else
      begin
        data_i = DateTime.strptime(params[:data_inicial], '%d/%m/%Y
%H:%M')
        data_f = DateTime.strptime(params[:data_final], '%d/%m/%Y %H:
%M')
        params[:data_inicial] != nil and params[:data_inicial] != nil
        @amostras = Amostra.where("created_at >= :start_date AND
created_at <= :end_date",
          {:start_date => data_i, :end_date => data_f })
      rescue
        @amostras = []
      end
  end
    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @amostras }
    end
end
```

Ilustração 8: método “index” “AmostrasController”.

Para evitar relacionamentos entre entidades que tiveram instâncias destruídas, o método “destroy”, antes de ser executado, foi modificado para verificar se a amostra é válida.

Pode não fazer muito sentido validar uma entidade antes de destruí-la, afinal validações são geralmente feitas antes de gravações no banco de dados, porém esta solução permitiu resolver o problema de instâncias que não podiam ser deletadas pois outras entidades dependiam de sua existência, para que não houvessem erros durante a usabilidade do sistema.

```
# DELETE /amostras/1
# DELETE /amostras/1.json
def destroy
    @amostra = Amostra.find(params[:id])
    if @amostra.valid?
      @amostra.destroy
        respond_to do |format|
        format.html { redirect_to amostras_url }
        format.json { head :ok }
      end
    else
      respond_to do |format|
        format.html { render action: "edit" }
        format.json { render json: @amostra.errors, status:
:unprocessable_entity }
      end
    end
end
```

Ilustração 9: método “destroy” “AmostrasController”.

Foi criado um método, “duplicarAmostra”, que como o próprio nome indica, serve para duplicar amostras quando o usuário necessita agilidade e exista já uma amostra parecida com a que ele pretende criar.

Apesar deste método funcionar, ele não foi mais utilizado, pois sua função deixou de ser necessária com a criação de novas facilidades do sistema.

```
# POST /amostras
# POST /amostras.json
def duplicarAmostra
    amostra = Amostra.find(params[:id])
    @amostra = Amostra.new
    @amostra.data = amostra.data
    @amostra.operacao = amostra.operacao
    @amostra.origem = amostra.origem
    @amostra.tipo_amostra = amostra.tipo_amostra
    @amostra.produto = amostra.produto
    @amostra.observacao = amostra.observacao
    #@amostra.certificado_Ilab = amostra.certificado_Ilab
    #@amostra.numero_gerado_Labware = amostra.numero_gerado_Labware
    @amostra.responsavel = @usuario
    respond_to do |format|
      if @amostra.save
        format.html { redirect_to @amostra, notice: 'Amostra criada
com sucesso.' }
        format.json { render json: @amostra, status: :created,
location: @amostra }
      else
        format.html { render action: "new" }
        format.json { render json: @amostra.errors, status:
:unprocessable_entity }
      end
    end
end
```

Ilustração 10: método “duplicarAmostra” “AmostrasController”.

7.2.2 Modificações em “class OperacaoComVagaosController” (Controlador)

A classe “OperacaoComVagaosController” teve seu método “create” modificado para atualizar o vagão correspondente com a operação criada de modo que seu status de carregamento indique verdadeiro, assim é permitido uma visualização de todos os vagões cadastrados para operar com seus respectivos indicativos de carregamento, facilitando posteriores validações.

```
# POST /operacao_com_vagaos
# POST /operacao_com_vagaos.json
def create
    @operacao_com_vagao =
OperacaoComVagao.new(params[:operacao_com_vagao])
    respond_to do |format|
      if @operacao_com_vagao.save
        @operacao_com_vagao.vt.update_attributes(:vt_carregado =>
true)
       format.html { redirect_to @operacao_com_vagao, notice:
'Operação com vagões criada com sucesso.' }
       format.json { render json: @operacao_com_vagao, status:
:created, location: @operacao_com_vagao }
      else
        format.html { render action: "new" }
        format.json { render json: @operacao_com_vagao.errors, status:
:unprocessable_entity }
      end
    end
end
```

Ilustração 11: método “create” “OperacaoComVagaosController”.

Também foi criado o método “limparComposição”. Este método tem o objetivo de retornar ao estado original de uma operação com vagão após ter sido criada através do lançamento de uma nata de vagão. Ele permite que seja desfeita uma associação errônea entre uma determinada operação com vagão e uma composição de vagão, retornando também o atributo “:vt_carregado” para verdadeiro do vagão associado.

```
def limparComposicao
    @operacao_com_vagao =
OperacaoComVagao.find(params[:operacao_com_vagao][:id])
    respond_to do |format|
      if @operacao_com_vagao.update_attributes(:composicao_de_vagao_id
=> nil)
        @operacao_com_vagao.vt.update_attributes(:vt_carregado =>
true)
        format.html { redirect_to @operacao_com_vagao, notice:
'Operação com vagão atualizada com sucesso.' }
        format.json { head :ok }
      else
        format.html { render action: "edit" }
        format.json { render json: @operacao_com_vagao.errors, status:
:unprocessable_entity }
      end
    end
end
```

Ilustração 12: método “limparComposição” “OperacaoComVagaosController”.

Para o método “destroy” a única alteração significativa foi atualizar o atributo do vagão associado para indicar que ele está descarregado antes de destruir a operação.

```
# DELETE /operacao_com_vagaos/1
# DELETE /operacao_com_vagaos/1.json
def destroy
    @operacao_com_vagao = OperacaoComVagao.find(params[:id])
    @operacao_com_vagao.vt.update_attributes(:vt_carregado => false)
    @operacao_com_vagao.destroy
    respond_to do |format|
      format.html { redirect_to operacao_com_vagaos_url }
      format.json { head :ok }
    end
end
```

Ilustração 13: método “destroy” “OperacaoComVagaosController”.

7.2.3 Modificações em “class AmostrasNavio” (Modelo)

A classe modelo “AmostraNavio” foi totalmente criada manualmente, ela não tem relação com nenhuma tabela do banco de dados, isto porque ela serve apenas para criar amostras em massa.

Como ela não se comunica com o banco de dados, ela não necessita herdar as características do “ActiveRecord” do Rails, responsável por esta comunicação. Para seu correto funcionamento basta ter as características de um “Model” Rails.

```
class AmostrasNavio

  include ActiveModel::Validations

  include ActiveModel::Conversion

  extend ActiveModel::Naming
```

Ilustração 14: “Model” “AmostraNavio”.

Como esta classe não se comunica com o banco de dados, seus atributos devem ser declarados como é mostrado.

```
attr_accessor :amostras, :data, :responsavel, :operacao, :produto,
:nome_navio, :viagem, :berco, :tanques, :observacao, :finalidades,
:lacres
```

Ilustração 15: atributos “AmostraNavio”.

Na sequência tem-se os métodos de validações presentes no Rails por padrão, as validações personalizadas (“validarAmostra” e “validarLacres”), a inicialização dos atributos

com valores pré-determinados e como não há necessidade de persistência no banco de dados, o método “persisted?” é setado como falso.

```
validates_presence_of :nome_navio, :viagem, :berco, :tanques,
:finalidades
  validate :validarAmostra
  validate :validarLacres
  def initialize(attributes = {:tanques => [], :finalidades => []})
    attributes.each do |name, value|
      send("#{name}=", value)
    end
  end
  def persisted?
    false
  end
```

Ilustração 16: métodos de validações “AmostraNavio”.

O método “criarAmostraNavio” foi criado para retornar uma amostra válida.

```
def criarAmostraNavio
    amostra = Amostra.new(:data => data,
    :responsavel_id => responsavel,
    :operacao_id => operacao,
    :produto_id => produto,
    :origem_id => 44,
    :tipo_amostra_id => 1)
    amostra
end
```

Ilustração 17: método “criarAmostraNavio” “AmostraNavio”.

Tendo o método “criarAmostraNavio”, é possível utilizar o método “grupoAmostras” para criar amostras em grupo. Basicamente seu funcionamento consiste em criar uma amostra para cada tanque “tq” inserido no *array* “tanques”.

```
def grupoAmostras
    amostras = []
    if tanques == []
      amostra =  criarAmostraNavio
      amostra.observacao = "Nome do navio / Vessel: #{nome_navio}.
Viagem / Voyage: #{viagem}. Berço / Berth: #{berco}. Tanque / Tank:
#{tanques}. Obs.: #{observacao}"
    amostras << amostra
    else
      tanques.each do |tq|
        amostra = criarAmostraNavio
        amostra.observacao = "Nome do navio / Vessel: #{nome_navio}.
Viagem / Voyage: #{viagem}. Berço / Berth: #{berco}. Tanque / Tank:
#{tq}. Obs.: #{observacao}"
        amostras << amostra
      end
    end
    amostras
  end
```

Ilustração 18: método “grupoAmostras” “AmostraNavio”.

De acordo com as necessidades dos usuários, os frascos de amostras que serão utilizados para análise, não necessitam de lacres. Sendo assim, o método “numeroEtiquetasComLacre” foi criado para contribuir com esta regra de negócio. Ele verifica se existe a necessidade do usuário em incluir a finalidade análise terminal e dependendo disto retorna o número de lacres necessários para o conjunto de amostras de navio. Sucintamente ele diz quantos lacres uma instância do modelo “AmostrasNavio” necessita de acordo com seus atributos internos.

```
def numeroEtiquetasComLacre
    if finalidades.include?("analise_terminal")
      etiquetas_com_lacre = tanques.length * (finalidades.length - 1)
    else
      etiquetas_com_lacre = tanques.length * finalidades.length
    end
    etiquetas_com_lacre
  end
```

Ilustração 19: método “numeroEtiquetasComLacre” “AmostraNavio”.

As validações, como são acessadas internamente, devem assumir comportamento privado, “private”.

O método de validação “validarAmostra” verifica dois atributos e faz duas validações, na primeira valida a amostra que retornou do método “criarAmostraNavio” e a segunda checa se alguma operação não permitida para este tipo de amostra foi celecionada e toma as ações necessárias caso ocorra.

```
private
  def validarAmostra
    if self.tanques == nil
      self.tanques = []
    end
    if self.finalidades == nil
      self.finalidades = []
    end
    amostra = criarAmostraNavio
    if !amostra.valid?
      amostra.errors.full_messages.each do |msg|
        errors.add :amostras, ": #{msg}"
      end
    end
    if ![1,2,10].include?(operacao.to_i)
      errors.add :operacao, ": Operação inválida para este tipo de
amostra"
    end
  end
```

Ilustração 20: método “validarAmostra” “AmostraNavio”.

Já o método “validarLacres”, valida se a quantidade de lacres inseridos está de acordo com a quantidade necessária e também valida se os lacres inserido estão válidos de acordo com o modelo dos lacres.

```
def validarLacres
    if lacres.split.length != 0
      lacres_array = lacres.split
      if lacres_array.length != numeroEtiquetasComLacre
        errors.add :lacres, ": Você está tentando incluir
#{lacres_array.length} lacre(s), quando o número requerido é
#{numeroEtiquetasComLacre}"
      end
      lacre_teste = Lacre.new(:amostra_id => Amostra.last.id,
:finalidade_id => Finalidade.last.id)
      lacres_array.each do |l|
        lacre_teste.numero = l
        if !lacre_teste.valid?
          lacre_teste.errors.full_messages.each do |msg|
            errors.add :lacres, ": #{l} #{msg}"
          end
        end
      end
      if lacres_array.uniq.length != lacres_array.length
        errors.add :lacres, ": Você está tentando incluir lacres com
números repetidos"
      end
    end
  end
  public
end
```

Ilustração 21: método “validarLacres” “AmostraNavio”.

CONSIDERAÇÕES FINAIS

A UFPR Litoral e a Transpetro foram os meios desencadeadores do processo de criação deste projeto. O âmbito acadêmico e profissional proporcionaram a oportunidade de contribuição para uma melhor forma de trabalho e aplicação dos conhecimentos e recursos adquiridos nesta vivencia.

Como o Projeto Político Pedagógico da UFPR Litoral visa incentivar a contribuição para o desenvolvimento tecno-científico e social da região, de forma prática e teórica, bem como o Projeto Político do Curso de Informática e Cidadania, estimula as práticas de criação e utilização de sistemas informatizados integrados com o uso e administração de sistemas operacionais, banco de dados, linguagens e técnicas de programação.

Os avanços na área de gestão e suas ferramentas cada vez mais informatizadas criam um caminho natural para meios digitais no controle das informações, sendo esta uma forma de dinamizar os processos de trabalho, organizando-os e criando bancos de dados a serem utilizados como informações para possíveis tomadas de decisões.

A utilização da WEB como ferramenta neste processo facilita o funcionamento e o alcance ao sistema, podendo ser acessado de qualquer lugar conectado a intranet da empresa e as informações são disponíveis apenas aos colaboradores devidamente cadastrados.

Atualmente o sistema objeto deste trabalho tem em sua base de dados mais de cinco mil e trezentos registros e está em funcionamento desde maio de 2012. No início houve certa resistência por parte dos usuários menos familiarizados com computador, afinal o processo veio como uma forma nova de trabalho, que exige algum domínio de uma tecnologia que até então não era bem compreendida por eles, porém após a fase inicial e aplicação de treinamentos, não existiu mais relatos negativos, mas sim novas demandas, que necessitariam de outros módulos que as atendessem.

É importante salientar que os treinamentos foram aplicados de forma prática aos usuários, diretamente no local de trabalho, facilitando assim a assimilação dos recursos. Este processo é contínuo e sofre renovações de acordo com as necessidades de cada indivíduo e da mesma forma estes agem como multiplicadores desta aprendizagem.

A informatização e implantação do sistema em questão proporcionou, além de reduzir drasticamente a ineficiência de procedimentos que não atingiam ou atingiam parcialmente as necessidades das tarefas, uma maior inclusão digital e tecnológica de trabalhadores que até então apresentavam resistência e preconceito para com estes processos e recursos. Resultando

em maior eficiência, contribuindo para a agilidade nos processos de controle de amostras e operações de vagões, exaurindo menos os trabalhadores, física e psicologicamente, antes envolvidos com tarefas manuais e repetitivas que geravam stress e não garantiam resultados confiáveis ou de fácil conferência e posterior consulta. Fato que poderia comprometer o nível de confiabilidade das informações geradas em caso de auditorias legais que resultariam em não conformidades podendo assim ocasionar prejuízos financeiros e danos jurídicos à empresa e possivelmente também aos empregados envolvidos.

Também é importante salientar que desde a implantação do sistema e após a fase de adaptação, foi possível a correção de erros pontuais ocasionados por colaboradores, o que resultou na observação de uma expressiva redução nas não conformidades em auditorias, tanto internas quanto externas, creditando assim uma maior seriedade no processo por parte dos envolvidos.

Também houveram relatos positivos para a nova metodologia e ferramentas utilizadas pela maioria dos usuários, devido a uma maior agilidade proporcionada. Um dos pontos altos neste sentido foi a redução de tempo para emissão de etiquetas em operações com navios, e a desburocratização da atividade.

Os objetivos de facilitar o armazenamento e cruzamento de dados e agilizar o acesso a informação, são naturalmente alcançados com a informatização e criação de um sistema dotado de banco de dados, mas para isto o sistema deve atender as necessidades específicas de cada usuário ou chegar o mais perto possível disto e a barreira contra a TI deve ser rompida. Um sistema de fácil usabilidade e intuitivo ajuda muito neste sentido.

Devido a necessidades internas, logo após a implantação do módulo de controle das operações com vagões, a gerência da empresa decidiu optar por não mais operar com o modal de vagões, o que tornou este módulo atualmente necessário somente para possíveis auditorias e consultas que por ventura possam ser necessárias, porém caso a empresa torne a usar esta operação, o sistema atenderá normalmente.

O sistema ainda necessita de alguns ajustes, como por exemplo a inserção do recurso de reinicialização automática após queda do servidor compartilhado, processo que está em andamento.

Como a área de TI é muito dinâmica, já foi possível observar novas formas de desenvolver com melhores resultados outras funcionalidades, através de novas tecnologias como banco de dados NoSql por exemplo, que se mostra promissor e quebra um paradigma antigo de estrutura de banco de dados com possibilidades de agilizar o processo de desenvolvimento em Back-End.

Simple Form e Twitter Bootstrap são outros bons exemplos de tecnologias encontradas para agilizar o processo, só que neste caso em Front-End.

O estudo e aplicação de técnicas de desenvolvimento ágil, também seriam úteis nos processos posteriores.

Gerando base de conhecimento para novos projetos, foi possível, através deste trabalho advindo do desenvolvimento do sistema e sua divulgação no meio acadêmico, o início do processo de criação de um novo sistema para o Centro Cultural da UFPR Litoral, este novo sistema teve seus alicerces criados com a forma de trabalho e tecnologias muito próximas das descritas neste trabalho.

REFERENCIAS

RICARTE, I. L. M. **Bancos de Dados Relacionais.** Disponível em: <http://www.dca.fee.unicamp.br/cursos/PooJava/javadb/bdrel.html> Acesso em: 04 de Agosto de 2013.

YIN, Robert K. Estudo de Caso: planejamento e métodos. 2. ed. Porto Alegre: Bookman, 2001.

REIS, Dácio Roberto dos. Gestão da Inovação Tecnológica. 2ªed. Barueri, SP: Manole ,2008

Ruby on Rails Brasil. Disponível em: <http://www.rubyonrails.com.br/> Acesso em: 25 de Março de 2012.

Apostila Desenv. Ágil para Web com Ruby on Rails - Caelum. Disponível em: <http://www.caelum.com.br/apostila-ruby-on-rails/> Acesso em: 15 de Agosto de 2012.

Ruby-Doc.org: Documenting the Ruby Language. Disponível em: <http://www.ruby-doc.org/> Acesso em: 02 de Outubro de 2011.

Dia a drawing program. Disponível em: <https://projects.gnome.org/dia/> Acesso em: 06 de Outubro de 2011.

Aptana. Disponível em: <http://www.aptana.com/> Acesso em: 19 de Março de 2012.

About SQLite. Disponível em: <http://www.sqlite.org/about.html> Acesso em: 08 de Janeiro de 2012.

Navegador Firefox em Português Brasileiro | Mais rápido, seguro e personalizável. Disponível em: <http://www.mozilla.org/pt-BR/firefox/fx/> Acesso em: 04 de Maio de 2011.

VirtualBox 4.2.18.88780 | Download | TechTudo. Disponível em: <http://www.techtudo.com.br/tudo-sobre/s/virtualbox.html> Acesso em: 22 de Julho de 2011.

O que é um framework?. Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm> Acesso em: 08 de Agosto de 2013.

O que é o Ubuntu? | Comunidade Ubuntu Brasil. Disponível em: <http://www.ubuntu-br.org/ubuntu> Acesso em: 08 de Agosto de 2013.

livros_java_x_ruby_on_rails.jpg (imagem JPEG, 400×330 pixels). Disponível em: <http://www.elvis.eti.br/wp-content/uploads/2008/09/livros_java_x_ruby_on_rails.jpg> Acesso em: 08 de Agosto de 2013.

Histórico | Educação é a nossa praia. Disponível em: <http://www.litoral.ufpr.br/historico> Acesso em: 01 de Setembro de 2013.

Projeto Político Pedagógico | Educação é a nossa praia. Disponível em: <http://www.litoral.ufpr.br/ppp> Acesso em: 01 de Setembro de 2013.

Projeto Político Pedagógico – UFPR – setor litoral - PPP - UFPR - litoral. Disponível em: <http://www.litoral.ufpr.br/sites/default/files/PPP%20-%20UFPR%20-%20LITORAL.pdf> Acesso em: 01 de Setembro de 2013.

MAZZOLA, V. B. *Engenharia de Software*. 145 p.

CAMARGO, N. F. *VFT: Uma estrutura de aplicação para visualização de fenômenos terrestres em 3d via web*. 2003. 182 f. Dissertação (Mestrado em Informática) – Universidade Federal do Paraná, Curitiba. 2003.

RailRoady :: Ruby Model & Controller Class Diagram Generator. Disponível em: <http://railroady.prestonlee.com/> Acesso em: 07 de Setembro de 2013.

CasosdeUsoAl.pdf. Disponível em: <http://www.inf.ufpr.br/silvia/ESNovo/UML/pdf/CasosdeUsoAl.pdf> Acesso em: 08 de Setembro de 2013.

8 APÊNDICE A: DIAGRAMAS GERADOS PELA GEM RAILROADY

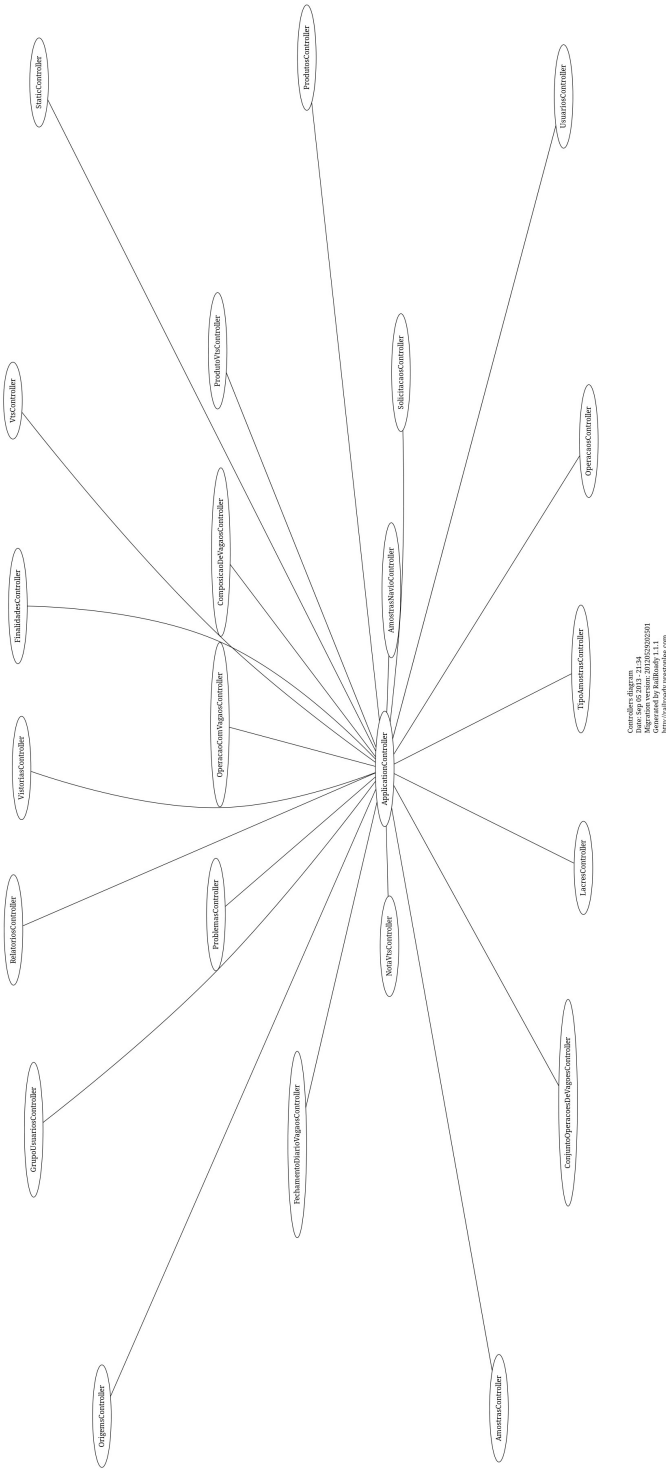


Ilustração 22: controllers_brief.

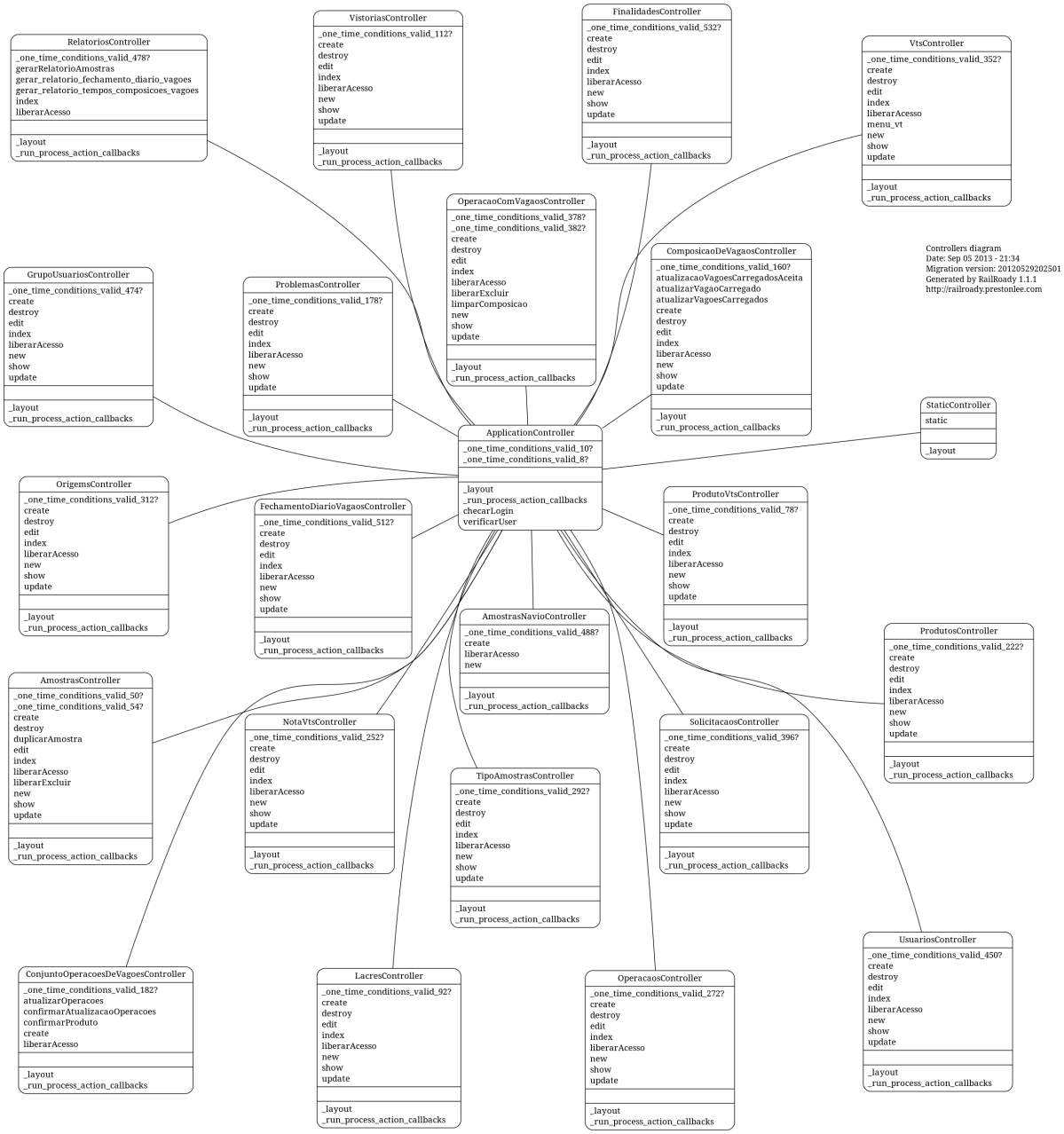


Ilustração 23: controllers_complete

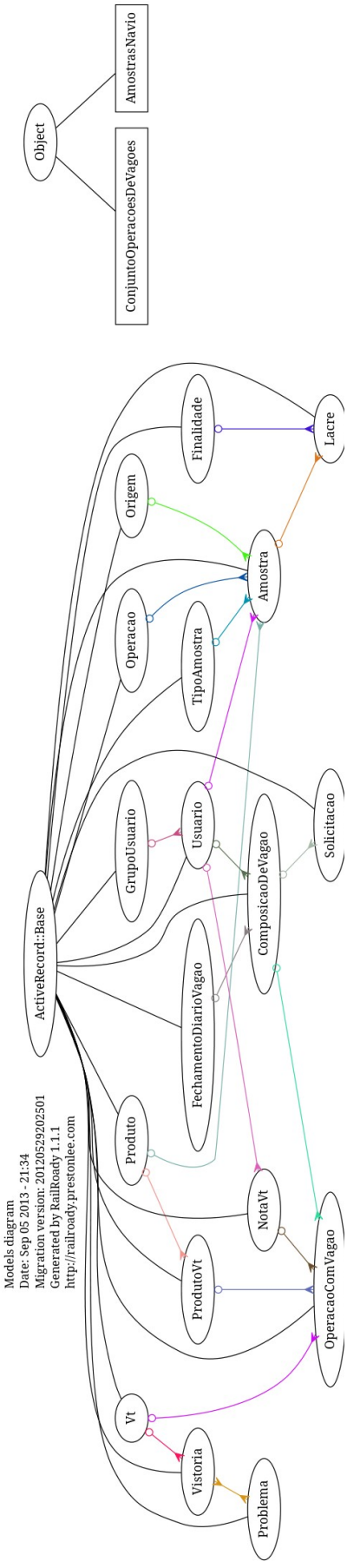


Ilustração 24: models_brief

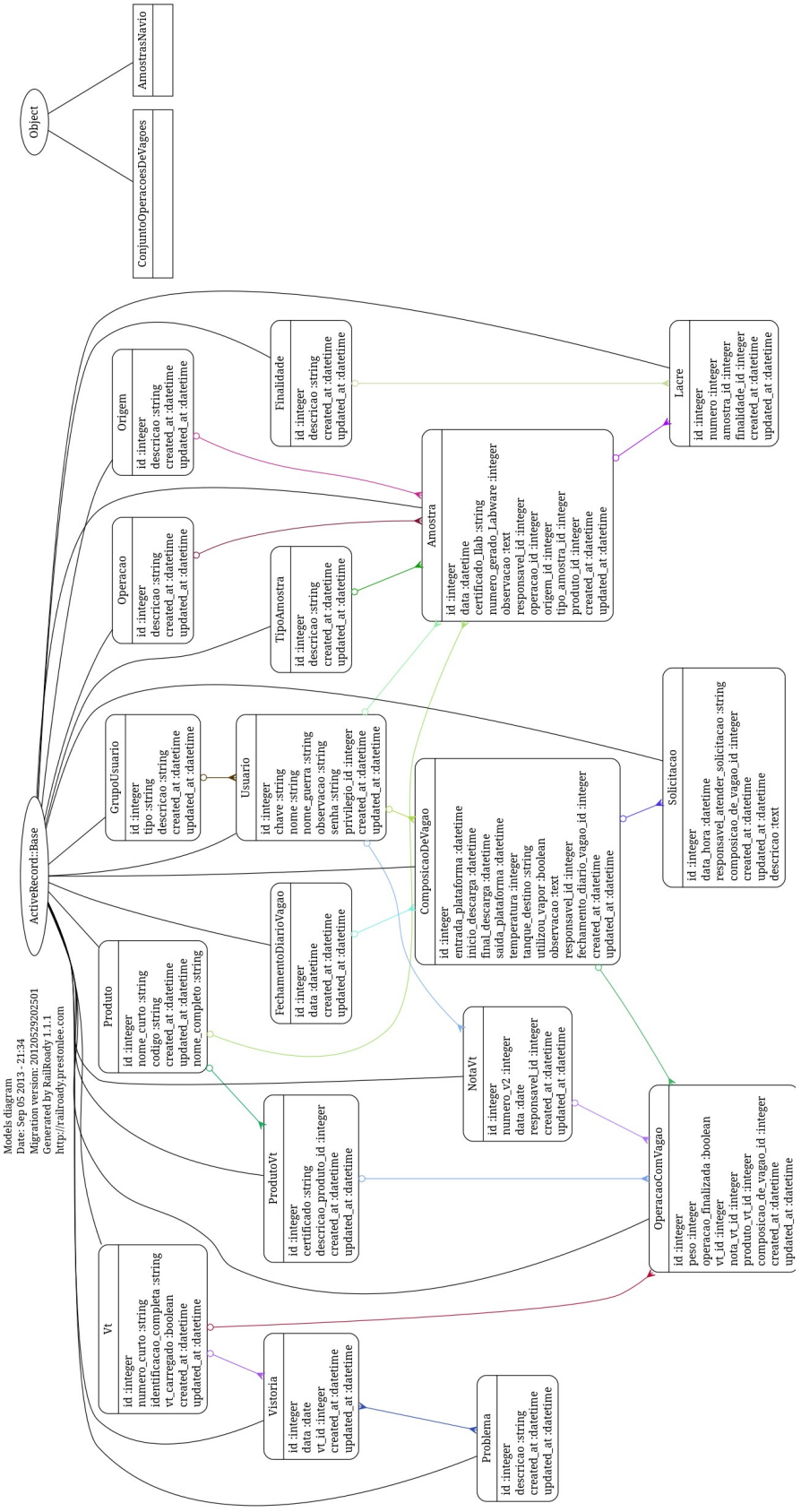


Ilustração 25: models_complete

9 APÊNDICE B: CÓDIGO FONTE

9.1 CÓDIGO GERADO PELO DIA ATRAVÉS DO EXPORTADOR PARA PASCAL, UTILIZADO COMO BASE PARA GERAR OS SCAFFOLD'S

```
/* generated by dia/codegen.py */

Type

 Amostra = class;

 Composicao_de_vagao = class;

 Fechamento_diario_vagao = class;

 Finalidade = class;

 Grupo_usuario = class;

 Lacre = class;

 Nota_vt = class;

 Operacao = class;

 Operacao_com_vagao = class;

 Origem = class;

 Problema = class;

 Produto = class;

 Produto_vt = class;

 Solicitacao = class;
```

```
Tipo_amostra = class;

Usuario = class;

Vistoria = class;

Vt = class;

problemas_vistorias = class;



Amostra = class

public

  data : datetime;

  certificado_Ilab : string;

  numero_gerado_Labware : integer;

  observacao : text;

  responsavel : references;

  operacao : references;

  origem : references;

  tipo_amostra : references;

  produto : references;

  Procedure isImprimirEtiqueta;

  Procedure imprimirEtiqueta;

End;
```

```
Composicao_de_vagao = class
public
  responsavel : references;
  inicio_descarga : datetime;
  final_descarga : datetime;
  saida_plataforma : datetime;
  entrada_plataforma : datetime;
  fechamento_diario_vagao : references;
  temperatura : integer;
  tanque_destino : string;
  utilizou_vapor : boolean;
  observacao : text;
End;

Fechamento_diario_vagao = class
public
  data : datetime;
End;
```

Finalidade = class

public

descricao : string;

End;

Grupo_usuario = class

public

tipo : string;

descricao : string;

End;

Lacre = class

public

numero : integer;

amostra : references;

finalidade : references;

End;

Nota_vt = class

public

```
  numero_de_controle : integer;

  data : date;

  responsavel : references;

End;


Operacao = class

public

  descricao : string;

End;


Operacao_com_vagao = class

private

  Procedure vtVazio;

public

  peso : integer;

  operacao_finalizada : boolean;

  vt : references;

  nota_vt : references;

  produto_vt : references;

  composicao_de_vagao : references;
```

```
  Procedure numeroVagao;
End;

Origem = class
public
  descricao : string;
End;

Problema = class
public
  descricao : string;
  Procedure vtsPorProblema(problema:);
End;

Produto = class
public
  descricao : string;
  codigo : string;
End;
```

Produto_vt = class

public

certificado : string;

descricao_produto : references;

Procedure nomeProduto;

Procedure produtoECertificado;

End;

Solicitacao = class

public

data_hora : datetime;

responsavel_atender_solicitacao : string;

composicao_de_vagao : references;

End;

Tipo_amostra = class

public

descricao : string;

End;

```
Usuario = class

public

  chave : string;

  nome : string;

  nome_guerra : string;

  observacao : string;

  senha : string;

  privilegio : references;

End;



Vistoria = class

public

  data : date;

  vt : references;

End;



Vt = class

private

  Procedure validaFormatoPeso(string:);

  Procedure validaFormatoVagao(string:);
```

```
public

  numero_curto : string;

  identificacao_completa : string;

  vt_carregado : boolean;

  Procedure validaStringVagoesPesos(string:;string:);

End;



problemas_vistorias = class

public

  problema : references;

  vistoria : references;

End;
```

9.2 CÓDIGO TRATADO COM O GEDIT DO UBUNTU PARA GERAR OS SCAFFOLD'S

```
rails g scaffold Amostra data:datetime certificado_Ilab:string numero_gerado_Labware:integer responsavel:references operacao:references origem:references tipo_amostra:references produto:references
rails g scaffold Finalidade descricao:string
rails g scaffold Lacre numero:integer amostra:references finalidade:references
rails g scaffold Grupo_usuario tipo:string descricao:string
rails g scaffold Nota_vt numero_de_controle:integer data:date responsavel:references
rails g scaffold Operacao descricao:string
```

rails g scaffold Composicao_de_vagao entrada_plataforma:datetime inicio_descarga:datetime final_descarga:datetime saida_plataforma:datetimeresponsavel:references fechamento_diario_vagao:references
rails g scaffold Fechamento_diario_vagao datahora:datetime
rails g scaffold Operacao_com_vt peso:integer vt_carregado:boolean vt:references nota_vt:references produto_vt:references composicao_de_vagao:references
rails g scaffold Origem descricao:string
rails g scaffold Problema descricao:string
rails g scaffold Produto descricao:string codigo:string
rails g scaffold Produto_vt certificado:string descricao_produto:references
rails g scaffold Solicitacao data_hora:datetime responsavel_atender_solicitacao:string composicao_de_vagao:references
rails g scaffold Tipo_amostra descricao:string
rails g scaffold Usuario chave:string nome:string nome_guerra:string observacao:string senha:string privilegio:references
rails g scaffold Vistoria data:date vt:references
rails g scaffold Vt numero_curto:string identificacao_completa:string

OBS. Este código pode apresentar diferenças em relação ao código fonte do sistema, visto que ele foi usado para gerar a base do sistema, após isto houveram alterações no código fonte para atender as necessidades das regras de negócio.

9.3 CÓDIGO FONTE DO “GEMFILE”

```
source 'http://rubygems.org'

gem 'rails', '3.1.0'

# Bundle edge Rails instead:
# gem 'rails',     :git => 'git://github.com/rails/rails.git'

gem 'sqlite3', '1.3.5'

#gem 'execjs'
gem 'therubyracer', '0.9.5'

#para gerar pdf
gem 'wkhtmltopdf-binary', '0.9.5.1'
gem 'pdfkit', '0.5.0'

#gerar documentação
```

```
gem 'railroady'


# Gems used only for assets and not required
# in production environments by default.
group :assets do
  gem 'sass-rails', "  ~> 3.1.0"
  gem 'coffee-rails', "~> 3.1.0"
  gem 'uglifier'
end

gem 'jquery-rails'

# Use unicorn as the web server
gem 'unicorn'

# Deploy with Capistrano
# gem 'capistrano'

# To use debugger
# gem 'ruby-debug19', :require => 'ruby-debug'

group :test do
  # Pretty printed test output
  gem 'turn', :require => false
end
```

9.4 CÓDIGO FONTE DO “SCHEMA.RB”

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended to check this file into your version control system.

ActiveRecord::Schema.define(:version => 20120529202501) do

  create_table "amostras", :force => true do |t|
    t.datetime "data"
    t.string   "certificado_Ilab"
    t.integer  "numero_gerado_Labware"
    t.text     "observacao"
    t.integer  "responsavel_id",       :null => false
    t.integer  "operacao_id",          :null => false
    t.integer  "origem_id",            :null => false
    t.integer  "tipo_amostra_id",      :null => false
    t.integer  "produto_id",           :null => false
```

```
  t.datetime "created_at"
  t.datetime "updated_at"
 end

 add_index "amostras", ["operacao_id"], :name => "index_amostras_on_operacao_id"
 add_index "amostras", ["origem_id"], :name => "index_amostras_on_origem_id"
 add_index "amostras", ["produto_id"], :name => "index_amostras_on_produto_id"
 add_index "amostras", ["responsavel_id"], :name => "index_amostras_on_responsavel_id"
 add_index "amostras", ["tipo_amostra_id"], :name => "index_amostras_on_tipo_amostra_id"

 create_table "composicao_de_vagaos", :force => true do |t|
  t.datetime "entrada_plataforma"
  t.datetime "inicio_descarga"
  t.datetime "final_descarga"
  t.datetime "saida_plataforma"
  t.integer  "temperatura"
  t.string   "tanque_destino"
  t.boolean  "utilizou_vapor"
  t.text     "observacao"
  t.integer  "responsavel_id",            :null => false
  t.integer  "fechamento_diario_vagao_id", :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
 end

 add_index "composicao_de_vagaos", ["fechamento_diario_vagao_id"], :name =>
"index_composicao_de_vagaos_on_fechamento_diario_vagao_id"
 add_index "composicao_de_vagaos", ["responsavel_id"], :name =>
"index_composicao_de_vagaos_on_responsavel_id"

 create_table "fechamento_diario_vagaos", :force => true do |t|
  t.datetime "data",       :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
 end

 create_table "finalidades", :force => true do |t|
  t.string   "descricao",  :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
 end

 create_table "grupo_usuarios", :force => true do |t|
  t.string   "tipo",       :null => false
  t.string   "descricao",  :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
 end

 create_table "lacres", :force => true do |t|
  t.integer  "numero"
  t.integer  "amostra_id",    :null => false
  t.integer  "finalidade_id", :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
 end

 add_index "lacres", ["amostra_id"], :name => "index_lacres_on_amostra_id"
 add_index "lacres", ["finalidade_id"], :name => "index_lacres_on_finalidade_id"
```

```
create_table "nota_vts", :force => true do |t|
  t.integer  "numero_v2",      :null => false
  t.date     "data",           :null => false
  t.integer  "responsavel_id", :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
end

add_index "nota_vts", ["responsavel_id"], :name => "index_nota_vts_on_responsavel_id"

create_table "operacao_com_vagaos", :force => true do |t|
  t.integer  "peso",                                :null => false
  t.boolean  "operacao_finalizada",    :default => false, :null => false
  t.integer  "vt_id",                               :null => false
  t.integer  "nota_vt_id",                          :null => false
  t.integer  "produto_vt_id",                       :null => false
  t.integer  "composicao_de_vagao_id"
  t.datetime "created_at"
  t.datetime "updated_at"
end

add_index "operacao_com_vagaos", ["composicao_de_vagao_id"], :name =>
"index_operacao_com_vagaos_on_composicao_de_vagao_id"
add_index "operacao_com_vagaos", ["nota_vt_id"], :name =>
"index_operacao_com_vagaos_on_nota_vt_id"
add_index "operacao_com_vagaos", ["produto_vt_id"], :name =>
"index_operacao_com_vagaos_on_produto_vt_id"
add_index "operacao_com_vagaos", ["vt_id"], :name => "index_operacao_com_vagaos_on_vt_id"

create_table "operacaos", :force => true do |t|
  t.string   "descricao",  :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
end

create_table "origems", :force => true do |t|
  t.string   "descricao",  :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
end

create_table "problemas", :force => true do |t|
  t.string   "descricao",  :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
end

create_table "problemas_vistorias", :id => false, :force => true do |t|
  t.integer  "problema_id"
  t.integer  "vistoria_id"
  t.datetime "created_at"
  t.datetime "updated_at"
end

create_table "produto_vts", :force => true do |t|
  t.string   "certificado",          :null => false
  t.integer  "descricao_produto_id", :null => false
  t.datetime "created_at"
  t.datetime "updated_at"
end
```

```
  add_index "produto_vts", ["descricao_produto_id"], :name =>
"index_produto_vts_on_descricao_produto_id"

  create_table "produtos", :force => true do |t|
    t.string   "nome_curto",    :null => false
    t.string   "codigo",        :null => false
    t.datetime "created_at"
    t.datetime "updated_at"
    t.string   "nome_completo"
  end

  create_table "solicitacaos", :force => true do |t|
    t.datetime "data_hora",                    :null => false
    t.string   "responsavel_atender_solicitacao", :null => false
    t.integer  "composicao_de_vagao_id",       :null => false
    t.datetime "created_at"
    t.datetime "updated_at"
    t.text     "descricao"
  end

  add_index "solicitacaos", ["composicao_de_vagao_id"], :name =>
"index_solicitacaos_on_composicao_de_vagao_id"

  create_table "tipo_amostras", :force => true do |t|
    t.string   "descricao",  :null => false
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  create_table "usuarios", :force => true do |t|
    t.string   "chave",         :null => false
    t.string   "nome",          :null => false
    t.string   "nome_guerra",   :null => false
    t.string   "observacao"
    t.string   "senha",         :null => false
    t.integer  "privilegio_id", :null => false
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  add_index "usuarios", ["privilegio_id"], :name => "index_usuarios_on_privilegio_id"

  create_table "vistorias", :force => true do |t|
    t.date     "data",       :null => false
    t.integer  "vt_id",      :null => false
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  add_index "vistorias", ["vt_id"], :name => "index_vistorias_on_vt_id"

  create_table "vts", :force => true do |t|
    t.string   "numero_curto",                :null => false
    t.string   "identificacao_completa",          :null => false
    t.boolean  "vt_carregado",          :default => false, :null => false
    t.datetime "created_at"
    t.datetime "updated_at"
  end
```

```
end
```

9.5 CÓDIGO FONTE DAS MIGRAÇÕES PARA O BANCO DE DADOS

```
class CreateProblemasVistorias < ActiveRecord::Migration
  # sem model somente tabela de ligação
  def change
    create_table :problemas_vistorias do |t|
      t.references :problema
      t.references :vistoria

      t.timestamps
    end
    add_index :problemas_vistorias, :problema_id
    add_index :problemas_vistorias, :vistoria_id
  end
end
```

```
class CreateAmostras < ActiveRecord::Migration
  def change
    create_table :amostras do |t|
      t.datetime :data
      t.string :certificado_Ilab
      t.integer :numero_gerado_Labware
      t.text :observacao
      t.references :responsavel, :null => false
      t.references :operacao, :null => false
      t.references :origem, :null => false
      t.references :tipo_amostra, :null => false
      t.references :produto, :null => false

      t.timestamps
    end
    add_index :amostras, :responsavel_id
    add_index :amostras, :operacao_id
    add_index :amostras, :origem_id
    add_index :amostras, :tipo_amostra_id
    add_index :amostras, :produto_id
  end
end
```

```
class CreateFinalidades < ActiveRecord::Migration
  def change
    create_table :finalidades do |t|
      t.string :descricao, :null => false
```

```
    t.timestamps
   end
  end
end
```

```
class CreateGrupoUsuarios < ActiveRecord::Migration
  def change
    create_table :grupo_usuarios do |t|
      t.string :tipo, :null => false
      t.string :descricao, :null => false

      t.timestamps
    end
  end
end
```

```
class CreateNotaVts < ActiveRecord::Migration
  def change
    create_table :nota_vts do |t|
      t.integer :numero_de_controle, :null => false
      t.date :data, :null => false
      t.references :responsavel, :null => false

      t.timestamps
    end
    add_index :nota_vts, :responsavel_id
  end
end
```

```
class CreateOperacaos < ActiveRecord::Migration
  def change
    create_table :operacaos do |t|
      t.string :descricao, :null => false

      t.timestamps
    end
  end
end
```

```
class CreateOperacaoComVagaos < ActiveRecord::Migration
  def change
    create_table :operacao_com_vagaos do |t|
      t.integer :peso, :null => false
      t.boolean :operacao_finalizada, :default => false, :null => false
```

```
    t.references :vt, :null => false
    t.references :nota_vt, :null => false
    t.references :produto_vt, :null => false
    t.references :composicao_de_vagao

    t.timestamps
   end
   add_index :operacao_com_vagaos, :vt_id
   add_index :operacao_com_vagaos, :nota_vt_id
   add_index :operacao_com_vagaos, :produto_vt_id
   add_index :operacao_com_vagaos, :composicao_de_vagao_id
  end
end




class CreateOrigems < ActiveRecord::Migration
  def change
   create_table :origems do |t|
    t.string :descricao, :null => false

    t.timestamps
   end
  end
end




class CreateProblemas < ActiveRecord::Migration
  def change
   create_table :problemas do |t|
    t.string :descricao, :null => false

    t.timestamps
   end
  end
end




class CreateProdutos < ActiveRecord::Migration
  def change
   create_table :produtos do |t|
    t.string :descricao, :null => false
    t.string :codigo, :null => false

    t.timestamps
   end
  end
end
```

```
class CreateProdutoVts < ActiveRecord::Migration
  def change
    create_table :produto_vts do |t|
      t.string :certificado, :null => false
      t.references :descricao_produto, :null => false

      t.timestamps
    end
    add_index :produto_vts, :descricao_produto_id
  end
end
```

```
class CreateTipoAmostras < ActiveRecord::Migration
  def change
    create_table :tipo_amostras do |t|
      t.string :descricao, :null => false

      t.timestamps
    end
  end
end
```

```
class CreateUsuarios < ActiveRecord::Migration
  def change
    create_table :usuarios do |t|
      t.string :chave, :null => false
      t.string :nome, :null => false
      t.string :nome_guerra, :null => false
      t.string :observacao
      t.string :senha, :null => false
      t.references :privilegio, :null => false

      t.timestamps
    end
    add_index :usuarios, :privilegio_id
  end
end
```

```
class CreateVistorias < ActiveRecord::Migration
  def change
    create_table :vistorias do |t|
      t.date :data, :null => false
      t.references :vt, :null => false

      t.timestamps
    end
    add_index :vistorias, :vt_id
```

```
  end
end
```

```
class CreateVts < ActiveRecord::Migration
  def change
    create_table :vts do |t|
      t.string :numero_curto, :null => false
      t.string :identificacao_completa, :null => false
      t.boolean :vt_carregado, :default => false, :null => false

      t.timestamps
    end
  end
end
```

```
class CreateSolicitacaos < ActiveRecord::Migration
  def change
    create_table :solicitacaos do |t|
      t.datetime :data_hora, :null => false
      t.string :responsavel_atender_solicitacao, :null => false
      t.references :composicao_de_vagao, :null => false

      t.timestamps
    end
    add_index :solicitacaos, :composicao_de_vagao_id
  end
end
```

```
class CreateLacres < ActiveRecord::Migration
  def change
    create_table :lacres do |t|
      t.integer :numero
      t.references :amostra, :null => false
      t.references :finalidade, :null => false

      t.timestamps
    end
    add_index :lacres, :amostra_id
    add_index :lacres, :finalidade_id
  end
end
```

```
class CreateComposicaoDeVagaos < ActiveRecord::Migration
  def change
```

```
    create_table :composicao_de_vagaos do |t|
      t.datetime :entrada_plataforma
      t.datetime :inicio_descarga
      t.datetime :final_descarga
      t.datetime :saida_plataforma
      t.integer  :temperatura
      t.string   :tanque_destino
      t.boolean  :utilizou_vapor
      t.text     :observacao
      t.references :responsavel, :null => false
      t.references :fechamento_diario_vagao, :null => false

      t.timestamps
    end
    add_index :composicao_de_vagaos, :responsavel_id
    add_index :composicao_de_vagaos, :fechamento_diario_vagao_id
  end
end




class CreateFechamentoDiarioVagaos < ActiveRecord::Migration
  def change
    create_table :fechamento_diario_vagaos do |t|
      t.datetime :data, :null => false

      t.timestamps
    end
  end
end




class AddDescricaoToSolicitacao < ActiveRecord::Migration
  def change
    add_column :solicitacaos, :descricao, :text
  end
end




class ChangeDescricaoFromProdutos < ActiveRecord::Migration
  def change
    change_table :produtos do |t|
      t.rename :descricao, :nome_curto
    end
  end
  add_column :produtos, :nome_completo, :string
end
```

```
class ChangeNumeroDeControleFromNotaVts < ActiveRecord::Migration
  def change
    change_table :nota_vts do |t|
      t.rename :numero_de_controle, :numero_v2
    end
  end
end
```

9.6 CÓDIGO FONTE DO ARQUIVO DE CONFIGURAÇÃO “PRODUTION.RB”

```
Sisteminha2::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb

  # Code is not reloaded between requests
  config.cache_classes = true

  # Full error reports are disabled and caching is turned on
  config.consider_all_requests_local       = false
  config.action_controller.perform_caching = true

  # Disable Rails's static asset server (Apache or nginx will already do this)
  # original ----> config.serve_static_assets = false
  #config.serve_static_assets = false
  config.serve_static_assets = true

  # Compress JavaScripts and CSS
  config.assets.compress = true

  # Don't fallback to assets pipeline if a precompiled asset is missed
  # original ----> config.assets.compile = false
  config.assets.compile = true
  #config.assets.compile = false

  # Generate digests for assets URLs
  config.assets.digest = true

  # Defaults to Rails.root.join("public/assets")
  # config.assets.manifest = YOUR_PATH

  # Specifies the header that your server uses for sending files
  # config.action_dispatch.x_sendfile_header = "X-Sendfile" # for apache
  # config.action_dispatch.x_sendfile_header = 'X-Accel-Redirect' # for nginx

  # Force all access to the app over SSL, use Strict-Transport-Security, and use secure cookies.
  # config.force_ssl = true

  # See everything in the log (default is :info)
  # config.log_level = :debug

  # Use a different logger for distributed setups
  # config.logger = SyslogLogger.new

  # Use a different cache store in production
  # config.cache_store = :mem_cache_store
```

```
  # Enable serving of images, stylesheets, and JavaScripts from an asset server
  # config.action_controller.asset_host = "http://assets.example.com"

  # Precompile additional assets (application.js, application.css, and all non-JS/CSS are already added)
  # config.assets.precompile += %w( search.js )

  # Disable delivery errors, bad email addresses will be ignored
  # config.action_mailer.raise_delivery_errors = false

  # Enable threaded mode
  # config.threadsafe!

  # Enable locale fallbacks for I18n (makes lookups for any locale fall back to
  # the I18n.default_locale when a translation can not be found)
  config.i18n.fallbacks = true

  # Send deprecation notices to registered listeners
  config.active_support.deprecation = :notify
end
```

9.7 CÓDIGO FONTE DOS CONTROLADORES

```
class AmostrasController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :new]
  before_filter :liberarExcluir, :only => [:destroy]

  def liberarAcesso
    if @usuario.privilegio.id == 2 or @usuario.privilegio.id == 3
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
  end

  def liberarExcluir
    if @usuario.privilegio.id == 3
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
  end

  # GET /amostras
  # GET /amostras.json
  def index
    if params[:data_inicial] == nil
      @amostras = Amostra.last(100)
    else
      begin
        data_i = DateTime.strptime(params[:data_inicial], '%d/%m/%Y %H:%M')
```

```
      data_f = DateTime.strptime(params[:data_final], '%d/%m/%Y %H:%M')
      params[:data_inicial] != nil and params[:data_inicial] != nil
      @amostras = Amostra.where("created_at >= :start_date AND created_at <= :end_date",
        {:start_date => data_i, :end_date => data_f })
    rescue
      @amostras = []
    end
  end

  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @amostras }
  end
end

# GET /amostras/1
# GET /amostras/1.json
def show
  @amostra = Amostra.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @amostra }
  end
end

# GET /amostras/new
# GET /amostras/new.json
def new
  @amostra = Amostra.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @amostra }
  end
end

# GET /amostras/1/edit
def edit
  @amostra = Amostra.find(params[:id])
end

# POST /amostras
# POST /amostras.json
def create
  @amostra = Amostra.new(params[:amostra])
  respond_to do |format|
    if @amostra.save
      #lacre = Lacre.new(:finalidade_id => 1, :amostra_id => @amostra.id)
      #lacre.save
      format.html { redirect_to @amostra, notice: 'Amostra criada com sucesso.' }
      format.json { render json: @amostra, status: :created, location: @amostra }
    else
      format.html { render action: "new" }
      format.json { render json: @amostra.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /amostras/1
```

```
  # PUT /amostras/1.json
  def update
    @amostra = Amostra.find(params[:id])
    respond_to do |format|
      if @amostra.update_attributes(params[:amostra])
        format.html { redirect_to @amostra, notice: 'Amostra atualizada com sucesso.' }
        format.json { head :ok }
      else
        format.html { render action: "edit" }
        format.json { render json: @amostra.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /amostras/1
  # DELETE /amostras/1.json
  def destroy
    @amostra = Amostra.find(params[:id])
    if @amostra.valid?
      @amostra.destroy

      respond_to do |format|
        format.html { redirect_to amostras_url }
        format.json { head :ok }
      end
    else
      respond_to do |format|
        format.html { render action: "edit" }
        format.json { render json: @amostra.errors, status: :unprocessable_entity }
      end
    end
  end

  # POST /amostras
  # POST /amostras.json
  def duplicarAmostra
    amostra = Amostra.find(params[:id])
    @amostra = Amostra.new
    @amostra.data = amostra.data
    @amostra.operacao = amostra.operacao
    @amostra.origem = amostra.origem
    @amostra.tipo_amostra = amostra.tipo_amostra
    @amostra.produto = amostra.produto
    @amostra.observacao = amostra.observacao
    #@amostra.certificado_Ilab = amostra.certificado_Ilab
    #@amostra.numero_gerado_Labware = amostra.numero_gerado_Labware
    @amostra.responsavel = @usuario

    respond_to do |format|
      if @amostra.save
        format.html { redirect_to @amostra, notice: 'Amostra criada com sucesso.' }
        format.json { render json: @amostra, status: :created, location: @amostra }
      else
        format.html { render action: "new" }
        format.json { render json: @amostra.errors, status: :unprocessable_entity }
      end
    end
  end

end
```

```
class AmostrasNavioController < ApplicationController

  before_filter :liberarAcesso, :except => [:new]
  def liberarAcesso
    if @usuario.privilegio.id == 2 or @usuario.privilegio.id == 3
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
  end

  # GET /amostras/new
  # GET /amostras/new.json
  def new
    @amostras_navio = AmostrasNavio.new
  end

  # POST /amostras
  # POST /amostras.json
  def create
    @amostras_navio = AmostrasNavio.new(params[:amostras_navio])
    respond_to do |format|
      if @amostras_navio.valid?
        lacres = @amostras_navio.lacres.split
        amostras = @amostras_navio.grupoAmostras

        amostras.each do |amostra|
          amostra.save
          if @amostras_navio.finalidades.include?("analise_terminal")
            lacre_an = Lacre.new(:amostra_id => amostra.id, :finalidade_id => 2)
            lacre_an.save
          end
          if @amostras_navio.finalidades.include?("arquivo_terminal")
            lacre_arq = Lacre.new(:amostra_id => amostra.id, :finalidade_id => 1)
            lacre_arq.numero = lacres.pop
            lacre_arq.save
          end
          if @amostras_navio.finalidades.include?("arquivo_navio")
            lacre_arqn = Lacre.new(:amostra_id => amostra.id, :finalidade_id => 3)
            lacre_arqn.numero = lacres.pop
            lacre_arqn.save
          end
          if @amostras_navio.finalidades.include?("repar")
            lacre_rep = Lacre.new(:amostra_id => amostra.id, :finalidade_id => 4)
            lacre_rep.numero = lacres.pop
            lacre_rep.save
          end
        end

        format.html { redirect_to "/amostras", notice: 'Amostras criadas com sucesso.' }
      else
        format.html { render action: "new" }
        format.json { render json: @amostras_navio.errors, status: :unprocessable_entity }
      end
```

```
  end
 end

end
```

```
require 'digest/sha2'
class ApplicationController < ActionController::Base

 before_filter :checarLogin

 protect_from_forgery

 private

 def checarLogin
  authenticate_or_request_with_http_basic('Login requerido! CHAVE e SENHA') do |username, password|
   verificarUser(username, password)
  end
 end

 def verificarUser(chave, senha)

  @usuario = Usuario.first(:conditions => ["chave = ? AND senha = ?", chave.to_s.upcase,
Digest::SHA2.hexdigest("linux #{senha} sudo su").force_encoding('UTF-8')])

 end

end
```

```
# encoding: utf-8
class ComposicaoDeVagaosController < ApplicationController

 before_filter :liberarAcesso, :except => [:index, :show, :new]
 def liberarAcesso
  if @usuario.privilegio.id == 2
  return true
  else
   redirect_to "/static/grupo_diferente"
  return false
  end
 end


 # GET /composicao_de_vagaos
 # GET /composicao_de_vagaos.json
 def index
  @composicao_de_vagaos = ComposicaoDeVagao.all

  respond_to do |format|
   format.html # index.html.erb
   format.json { render json: @composicao_de_vagaos }
  end
```

```
end

# GET /composicao_de_vagaos/1
# GET /composicao_de_vagaos/1.json
def show
  @composicao_de_vagao = ComposicaoDeVagao.find(params[:id])
  @composicao_de_vagao.conjunto_operacoes_de_vagoes = ConjuntoOperacoesDeVagoes.new

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @composicao_de_vagao }
  end
end

# GET /composicao_de_vagaos/new
# GET /composicao_de_vagaos/new.json
def new
  @composicao_de_vagao = ComposicaoDeVagao.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @composicao_de_vagao }
  end
end

# GET /composicao_de_vagaos/1/edit
def edit
  @composicao_de_vagao = ComposicaoDeVagao.find(params[:id])
end

# POST /composicao_de_vagaos
# POST /composicao_de_vagaos.json
def create
  @composicao_de_vagao = ComposicaoDeVagao.new(params[:composicao_de_vagao])
  respond_to do |format|
    if @composicao_de_vagao.save

      Email.enviarRelatorioTemposComposicoesVagoes.deliver
      format.html { redirect_to @composicao_de_vagao, notice: 'Composição de vagão criada com sucesso.' }
      format.json { render json: @composicao_de_vagao, status: :created, location: @composicao_de_vagao }
    else
      format.html { render action: "new" }
      format.json { render json: @composicao_de_vagao.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /composicao_de_vagaos/1
# PUT /composicao_de_vagaos/1.json
def update
  @composicao_de_vagao = ComposicaoDeVagao.find(params[:id])

  respond_to do |format|
    if @composicao_de_vagao.update_attributes(params[:composicao_de_vagao])
      if @composicao_de_vagao.final_descarga != nil
        @composicao_de_vagao.operacao_com_vagaos.each do |op|
          op.update_attributes(:operacao_finalizada => true)
          op.vt.update_attributes(:vt_carregado => false)
        end
      end
```

```
      format.html { redirect_to @composicao_de_vagao, notice: "Composição de vagão atualizada com
sucesso." }
      format.json { head :ok }
     else
      format.html { render action: "edit" }
      format.json { render json: @composicao_de_vagao.errors, status: :unprocessable_entity }
     end
   end
  end

  # DELETE /composicao_de_vagaos/1
  # DELETE /composicao_de_vagaos/1.json
  def destroy
   @composicao_de_vagao = ComposicaoDeVagao.find(params[:id])
   respond_to do |format|

    if @composicao_de_vagao.operacao_com_vagaos.empty?()

     @composicao_de_vagao.destroy
     format.html { redirect_to composicao_de_vagaos_url }
     format.json { head :ok }
    else

     format.html { redirect_to @composicao_de_vagao, alert: "ERRO: Não foi possível excluir a composição
de vagões
       pois existem operacões com vagões que pertencem a ela." }
    end

   end
  end

  def atualizarVagaoCarregado
   erros = []
   operacao_com_vagao = OperacaoComVagao.find(:first, :conditions =>
   ["vt_id  = ? AND operacao_finalizada = ? ", params[:Vt][:vt_id], false] )

   @composicao_de_vagao = ComposicaoDeVagao.find(params[:composicao_de_vagao_id])
   if operacao_com_vagao != nil
    if operacao_com_vagao.composicao_de_vagao == nil
     if operacao_com_vagao.update_attribute(:composicao_de_vagao, @composicao_de_vagao)
      flash[:notice] = "Operação com vagão atualizada com sucesso."
     else
      erros << "ERRO: Operação com vagão #{operacao_com_vagao.vt.numero_curto}
     não pôde ser atualizada no banco de dados."
     end
    else
     erros << "ERRO: Operação com vagão #{operacao_com_vagao.vt.numero_curto} não pode ser salva.
      É provável que ele já pertença a uma composição."
    end

   end

   if params[:Vt][:vt_id] != "" and operacao_com_vagao == nil
    erros << "ERRO: Operação com vagão não pode ser salva. Verifique se este vagão já não está com todas
as suas operações
          finalizadas. É provável que ele não esteja carregado."
   end

   respond_to do |format|
    format.html { redirect_to @composicao_de_vagao, alert: erros }
```

```
    end
  end

  def atualizarVagoesCarregados
    erros = []
    @composicao_de_vagao = ComposicaoDeVagao.find(params[:composicao_de_vagao_id])
    flash[:operacoes_com_vagoes] = params[:operacoes_com_vagoes]
    objVt = Vt.new
    vts = objVt.validaStringVagoesPesos(params[:operacoes_com_vagoes], "")
    array_numero_curto_vagoes = vts[1]
    vts= vts[0]
    vts_db = Vt.all(:conditions => ["numero_curto  IN (?) AND vt_carregado IN (?)",
      array_numero_curto_vagoes, [true]*array_numero_curto_vagoes.length])

    id_vts = []
    vts_db.each do |vt|
      id_vts << vt.id
    end

    #operacao_com_vagoes_db = OoperacaoComVagao.all(:conditions => ["operacao_finalizada = (?)",
false])
    operacao_com_vagoes_db = OperacaoComVagao.all(:conditions =>
    ["vt_id  IN (?) AND operacao_finalizada IN (?)", id_vts, [false]*id_vts.length])

    vts = objVt.procurarVagoesEmArray(vts, vts_db, "ERRO: Não existe uma operação com o vagão (", ") para
ser descarregada.")

    vts.each do |vt|
      vt[:erro][1] = nil
      erros = erros + vt[:erro]
    end

    operacao_com_vagoes_db.each do |operacao_com_vagao|
      if operacao_com_vagao.composicao_de_vagao != nil
        erros << "ERRO: Operação com o vagão #{operacao_com_vagao.vt.numero_curto} já pertence a uma
composição."
      end
    end

    erros  = erros - [nil]

    if erros == [] and params[:operacoes_com_vagoes] != ""
      @operacao_com_vagoes_db = [operacao_com_vagoes_db, @composicao_de_vagao]
      render "/grupo_op_vagoes/confirmar_grupo"
    else
      respond_to do |format|
        format.html { redirect_to @composicao_de_vagao, alert: erros }
      end
    end
  end

  def atualizacaoVagoesCarregadosAceita
    @composicao_de_vagao = ComposicaoDeVagao.find(params[:composicao_de_vagao_id])
    @operacao_com_vagaos = params[:operacao_com_vagaos]
    @operacao_com_vagaos.each do |id|
      OperacaoComVagao.find(id).update_attribute(:composicao_de_vagao, @composicao_de_vagao)
    end

    respond_to do |format|
```

```
    format.html { redirect_to @composicao_de_vagao, notice: 'Operações com vagões atualizadas com
sucesso.' }
   end
  end

end
```

```
#encoding: utf-8
class ConjuntoOperacoesDeVagoesController < ApplicationController

  before_filter :liberarAcesso, :except => []
  def liberarAcesso
   if @usuario.privilegio.id == 2
   return true
   else
    redirect_to "/static/grupo_diferente"
   return false
   end
  end

  def confirmarProduto
   @conjunto_operacoes_de_vagoes =
ConjuntoOperacoesDeVagoes.new(params[:conjunto_operacoes_de_vagoes])
   render "/conjunto_operacoes_de_vagoes/confirmar_produto"
  end

  def confirmarAtualizacaoOperacoes
   @conjunto_operacoes_de_vagoes =
ConjuntoOperacoesDeVagoes.new(params[:conjunto_operacoes_de_vagoes])
   @composicao_de_vagao =
ComposicaoDeVagao.find(@conjunto_operacoes_de_vagoes.composicao_de_vagao)
   @conjunto_operacoes_de_vagoes.composicao_de_vagao = @composicao_de_vagao
   @conjunto_operacoes_de_vagoes.pesos = "0"
   @conjunto_operacoes_de_vagoes.produto = "0"
   @conjunto_operacoes_de_vagoes.produto_para_confirmar = "0"

   if @conjunto_operacoes_de_vagoes.valid?
    @conjunto_operacoes_de_vagoes = @conjunto_operacoes_de_vagoes
    render "/conjunto_operacoes_de_vagoes/confirmar_atualizacao_operacoes"
   else
    @composicao_de_vagao.conjunto_operacoes_de_vagoes = @conjunto_operacoes_de_vagoes
    render "/composicao_de_vagaos/show"
   end
  end

  def atualizarOperacoes

   @conjunto_operacoes_de_vagoes =
ConjuntoOperacoesDeVagoes.new(params[:conjunto_operacoes_de_vagoes])
   @composicao_de_vagao =
ComposicaoDeVagao.find(@conjunto_operacoes_de_vagoes.composicao_de_vagao)
   @conjunto_operacoes_de_vagoes.op_vts.each do |op_vt|
    OperacaoComVagao.find(op_vt).update_attributes(:composicao_de_vagao_id =>
@composicao_de_vagao.id)
   end
   respond_to do |format|
```

```
    format.html { redirect_to @composicao_de_vagao, notice: 'Operações com vagões atualizadas com
sucesso.' }
   end

  end

  # POST /conjunto_operacoes_de_vagoes
  # POST /conjunto_operacoes_de_vagoes.json
  def create
   flash[:notice] =  nil
   @conjunto_operacoes_de_vagoes =
ConjuntoOperacoesDeVagoes.new(params[:conjunto_operacoes_de_vagoes])
   @nota_vt = NotaVt.find(@conjunto_operacoes_de_vagoes.nota_vt)
   @conjunto_operacoes_de_vagoes.nota_vt = @nota_vt
   if @conjunto_operacoes_de_vagoes.valid?
    @conjunto_operacoes_de_vagoes.op_vts.each do |op|
     op.save
     op.vt.update_attributes(:vt_carregado => true)
    end
    flash[:notice] =  'Operações com vagões criadas com sucesso.'
   end
   @nota_vt.conjunto_operacoes_de_vagoes = @conjunto_operacoes_de_vagoes
   render "/nota_vts/show"
  end

end




#encoding: utf-8
class FechamentoDiarioVagaosController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :new]
  def liberarAcesso
   if @usuario.privilegio.id == 2
   return true
   else
    redirect_to "/static/grupo_diferente"
   return false
   end
  end


  # GET /fechamento_diario_vagaos
  # GET /fechamento_diario_vagaos.json
  def index
   @fechamento_diario_vagaos = FechamentoDiarioVagao.last(50)

   respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @fechamento_diario_vagaos }
   end
  end

  # GET /fechamento_diario_vagaos/1
  # GET /fechamento_diario_vagaos/1.json
  def show
   @fechamento_diario_vagao = FechamentoDiarioVagao.find(params[:id])
```

```
    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @fechamento_diario_vagao }
    end
  end

  # GET /fechamento_diario_vagaos/new
  # GET /fechamento_diario_vagaos/new.json
  def new
    @fechamento_diario_vagao = FechamentoDiarioVagao.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @fechamento_diario_vagao }
    end
  end

  # GET /fechamento_diario_vagaos/1/edit
  def edit
    @fechamento_diario_vagao = FechamentoDiarioVagao.find(params[:id])
  end

  # POST /fechamento_diario_vagaos
  # POST /fechamento_diario_vagaos.json
  def create
    @fechamento_diario_vagao = FechamentoDiarioVagao.new(params[:fechamento_diario_vagao])

    respond_to do |format|
      if @fechamento_diario_vagao.save
        format.html { redirect_to @fechamento_diario_vagao, notice: 'Fechamento diário de vagões criado com
sucesso.' }
        format.json { render json: @fechamento_diario_vagao, status: :created, location:
@fechamento_diario_vagao }
      else
        format.html { render action: "new" }
        format.json { render json: @fechamento_diario_vagao.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /fechamento_diario_vagaos/1
  # PUT /fechamento_diario_vagaos/1.json
  def update
    @fechamento_diario_vagao = FechamentoDiarioVagao.find(params[:id])

    respond_to do |format|
      if @fechamento_diario_vagao.update_attributes(params[:fechamento_diario_vagao])
        format.html { redirect_to @fechamento_diario_vagao, notice: 'Fechamento diário de vagões atualizado
com sucesso.' }
        format.json { head :ok }
      else
        format.html { render action: "edit" }
        format.json { render json: @fechamento_diario_vagao.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /fechamento_diario_vagaos/1
  # DELETE /fechamento_diario_vagaos/1.json
```

```
  def destroy
    @fechamento_diario_vagao = FechamentoDiarioVagao.find(params[:id])
    respond_to do |format|
      if @fechamento_diario_vagao.composicao_de_vagaos.empty?()
        @fechamento_diario_vagao.destroy
        format.html { redirect_to fechamento_diario_vagaos_url }
        format.json { head :ok }
      else
        format.html { redirect_to @fechamento_diario_vagao, alert: ['ERRO: Não foi possível excluir o
fechamento diário de vagões pois
          existem composições de vagões que pertencem a ele.'] }
      end
    end
  end
end
```

```
# encoding: UTF-8
class FinalidadesController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :new]
  def liberarAcesso
    if @usuario.privilegio.id == 1
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
  end


  # GET /finalidades
  # GET /finalidades.json
  def index
    @finalidades = Finalidade.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @finalidades }
    end
  end

  # GET /finalidades/1
  # GET /finalidades/1.json
  def show
    @finalidade = Finalidade.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @finalidade }
    end
  end

  # GET /finalidades/new
  # GET /finalidades/new.json
  def new
    @finalidade = Finalidade.new
```

```
  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @finalidade }
  end
end

# GET /finalidades/1/edit
def edit
  @finalidade = Finalidade.find(params[:id])
end

# POST /finalidades
# POST /finalidades.json
def create
  @finalidade = Finalidade.new(params[:finalidade])

  respond_to do |format|
    if @finalidade.save
      format.html { redirect_to @finalidade, notice: 'Finalidade criada com sucesso.' }
      format.json { render json: @finalidade, status: :created, location: @finalidade }
    else
      format.html { render action: "new" }
      format.json { render json: @finalidade.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /finalidades/1
# PUT /finalidades/1.json
def update
  @finalidade = Finalidade.find(params[:id])

  respond_to do |format|
    if @finalidade.update_attributes(params[:finalidade])
      format.html { redirect_to @finalidade, notice: 'Finalidade atualizada com sucesso.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
      format.json { render json: @finalidade.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /finalidades/1
# DELETE /finalidades/1.json
def destroy
  @finalidade = Finalidade.find(params[:id])
  respond_to do |format|
    if @finalidade.lacres.empty?()
      @finalidade.destroy
      format.html { redirect_to finalidades_url }
      format.json { head :ok }
    else
      format.html { redirect_to @finalidade, alert: ['ERRO: Não foi possível excluir a finalidade pois existem
lacres que pertencem a ela.'] }
    end
  end
end
```

```
end



#encoding: utf-8
class GrupoUsuariosController < ApplicationController

  before_filter :liberarAcesso, :except => []
  def liberarAcesso
    if @usuario.id == 1
    return true
    else
      redirect_to "/404"
    return false
    end
  end


  # GET /grupo_usuarios
  # GET /grupo_usuarios.json
  def index
    @grupo_usuarios = GrupoUsuario.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @grupo_usuarios }
    end
  end

  # GET /grupo_usuarios/1
  # GET /grupo_usuarios/1.json
  def show
    @grupo_usuario = GrupoUsuario.find(params[:id])
    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @grupo_usuario }
    end
  end

  # GET /grupo_usuarios/new
  # GET /grupo_usuarios/new.json
  def new
    @grupo_usuario = GrupoUsuario.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @grupo_usuario }
    end
  end

  # GET /grupo_usuarios/1/edit
  def edit
    @grupo_usuario = GrupoUsuario.find(params[:id])
  end

  # POST /grupo_usuarios
  # POST /grupo_usuarios.json
  def create
```

```
    @grupo_usuario = GrupoUsuario.new(params[:grupo_usuario])

    respond_to do |format|
      if @grupo_usuario.save
        format.html { redirect_to @grupo_usuario, notice: 'Grupo de usuários criado com sucesso.' }
        format.json { render json: @grupo_usuario, status: :created, location: @grupo_usuario }
      else
        format.html { render action: "new" }
        format.json { render json: @grupo_usuario.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /grupo_usuarios/1
  # PUT /grupo_usuarios/1.json
  def update
    @grupo_usuario = GrupoUsuario.find(params[:id])

    respond_to do |format|
      if @grupo_usuario.update_attributes(params[:grupo_usuario])
        format.html { redirect_to @grupo_usuario, notice: 'Grupo de usuários atualizado com sucesso.' }
        format.json { head :ok }
      else
        format.html { render action: "edit" }
        format.json { render json: @grupo_usuario.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /grupo_usuarios/1
  # DELETE /grupo_usuarios/1.json
  def destroy
    @grupo_usuario = GrupoUsuario.find(params[:id])
    respond_to do |format|
      if @grupo_usuario.usuarios.empty?()
        @grupo_usuario.destroy
        format.html { redirect_to grupo_usuarios_url }
        format.json { head :ok }
      else
        format.html { redirect_to @grupo_usuario, alert: ['ERRO: Não foi possível excluir o grupo de usuários
pois existem usuários que pertencem a ele.'] }
      end
    end
  end
end




class LacresController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :new]
  def liberarAcesso
    if @usuario.privilegio.id == 2 or @usuario.privilegio.id == 3
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
```

```
end


# GET /lacres
# GET /lacres.json
def index
  @lacres = Lacre.all

  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @lacres }
  end
end

# GET /lacres/1
# GET /lacres/1.json
def show
  @lacre = Lacre.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @lacre }
  end
end

# GET /lacres/new
# GET /lacres/new.json
def new
  @lacre = Lacre.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @lacre }
  end
end

# GET /lacres/1/edit
def edit
  @lacre = Lacre.find(params[:id])
end

# POST /lacres
# POST /lacres.json
def create
  @lacre = Lacre.new(params[:lacre])

  respond_to do |format|
    if @lacre.save
      format.html { redirect_to @lacre, notice: 'Lacre criado com sucesso.' }
      format.json { render json: @lacre, status: :created, location: @lacre }
    else
      format.html { render action: "new" }
      format.json { render json: @lacre.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /lacres/1
# PUT /lacres/1.json
def update
```

```
    @lacre = Lacre.find(params[:id])

    respond_to do |format|
      if @lacre.update_attributes(params[:lacre])
        format.html { redirect_to @lacre, notice: 'Lacre atualizado com sucesso.' }
        format.json { head :ok }
      else
        format.html { render action: "edit" }
        format.json { render json: @lacre.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /lacres/1
  # DELETE /lacres/1.json
  def destroy
    @lacre = Lacre.find(params[:id])
    @lacre.destroy

    respond_to do |format|
      format.html { redirect_to lacres_url }
      format.json { head :ok }
    end
  end
end




#encoding: utf-8
class NotaVtsController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :new]
  def liberarAcesso
    if @usuario.privilegio.id == 2
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
  end

  # GET /nota_vts
  # GET /nota_vts.json
  def index

    if params[:data_inicial] == nil
      @nota_vts = NotaVt.last(100)
    else
      begin
        data_i = DateTime.strptime(params[:data_inicial], '%d/%m/%Y')
        data_f = DateTime.strptime(params[:data_final], '%d/%m/%Y')
        params[:data_inicial] != nil and params[:data_final] != nil
        @nota_vts = NotaVt.where(
          "data >= :start_date AND data <= :end_date",
          {:start_date => data_i, :end_date => data_f })
      rescue
        @nota_vts = []
      end
```

```
  end

  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @nota_vts }
  end
end

# GET /nota_vts/1
# GET /nota_vts/1.json
def show
  @nota_vt = NotaVt.find(params[:id])
  @nota_vt.conjunto_operacoes_de_vagoes = ConjuntoOperacoesDeVagoes.new
  @nota_vt.conjunto_operacoes_de_vagoes.nota_vt = @nota_vt

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @nota_vt }
  end
end

# GET /nota_vts/new
# GET /nota_vts/new.json
def new
  @nota_vt = NotaVt.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @nota_vt }
  end
end

# GET /nota_vts/1/edit
def edit
  @nota_vt = NotaVt.find(params[:id])
end

# POST /nota_vts
# POST /nota_vts.json
def create
  @nota_vt = NotaVt.new(params[:nota_vt])

  respond_to do |format|
    if @nota_vt.save
      format.html { redirect_to @nota_vt, notice: 'Nota de vagões criada com sucesso.' }
      format.json { render json: @nota_vt, status: :created, location: @nota_vt }
    else
      format.html { render action: "new" }
      format.json { render json: @nota_vt.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /nota_vts/1
# PUT /nota_vts/1.json
def update
  @nota_vt = NotaVt.find(params[:id])
  respond_to do |format|

    if @nota_vt.update_attributes(params[:nota_vt])
```

```
      format.html { redirect_to @nota_vt, notice: 'Nota de vagões atualizada com sucesso.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
      format.json { render json: @nota_vt.errors, status: :unprocessable_entity }
    end
  end
 end

 # DELETE /nota_vts/1
 # DELETE /nota_vts/1.json
 def destroy
  @nota_vt = NotaVt.find(params[:id])
  respond_to do |format|
    if @nota_vt.operacao_com_vagaos.empty?()
      @nota_vt.destroy
      format.html { redirect_to nota_vts_url }
      format.json { head :ok }
    else
      format.html { redirect_to @nota_vt, alert: ['ERRO: Não foi possível excluir a nota
        de vagões pois existem operações com vagões que utilizam ela.'] }
    end
  end
 end

end




# encoding: utf-8

class OperacaoComVagaosController < ApplicationController

 before_filter :liberarAcesso, :except => [:index, :show, :new]
 before_filter :liberarExcluir, :only => [:destroy]

 def liberarAcesso
  if @usuario.privilegio.id == 2
  return true
  else
    redirect_to "/static/grupo_diferente"
  return false
  end
 end

 def liberarExcluir
  if @usuario.privilegio.id == 1
  return true
  else
    redirect_to "/static/grupo_diferente"
  return false
  end
 end

 # GET /operacao_com_vagaos
 # GET /operacao_com_vagaos.json
 def index
```

```
  if params[:data_inicial] == nil
    @operacao_com_vagaos = OperacaoComVagao.last(300)
  else
    begin
      data_i = DateTime.strptime(params[:data_inicial], '%d/%m/%Y %H:%M')
      data_f = DateTime.strptime(params[:data_final], '%d/%m/%Y %H:%M')
      params[:data_inicial] != nil and params[:data_inicial] != nil
      @operacao_com_vagaos = OperacaoComVagao.where(
        "created_at >= :start_date AND created_at <= :end_date",
        {:start_date => data_i, :end_date => data_f })
    rescue
      @operacao_com_vagaos = []
    end
  end

  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @operacao_com_vagaos }
  end
end

# GET /operacao_com_vagaos/1
# GET /operacao_com_vagaos/1.json
def show
  @operacao_com_vagao = OperacaoComVagao.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @operacao_com_vagao }
  end
end

# GET /operacao_com_vagaos/new
# GET /operacao_com_vagaos/new.json
def new
  @operacao_com_vagao = OperacaoComVagao.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @operacao_com_vagao }
  end
end

# GET /operacao_com_vagaos/1/edit
def edit
  @operacao_com_vagao = OperacaoComVagao.find(params[:id])
end

# POST /operacao_com_vagaos
# POST /operacao_com_vagaos.json
def create
  @operacao_com_vagao = OperacaoComVagao.new(params[:operacao_com_vagao])
  respond_to do |format|
    if @operacao_com_vagao.save
      @operacao_com_vagao.vt.update_attributes(:vt_carregado => true)
      format.html { redirect_to @operacao_com_vagao, notice: 'Operação com vagões criada com sucesso.' }
      format.json { render json: @operacao_com_vagao, status: :created, location: @operacao_com_vagao }
    else
      format.html { render action: "new" }
      format.json { render json: @operacao_com_vagao.errors, status: :unprocessable_entity }
```

```
    end
  end
end

# PUT /operacao_com_vagaos/1
# PUT /operacao_com_vagaos/1.json
def update
  @operacao_com_vagao = OperacaoComVagao.find(params[:id])

  respond_to do |format|
    if @operacao_com_vagao.update_attributes(params[:operacao_com_vagao])
      format.html { redirect_to @operacao_com_vagao, notice: 'Operação com vagões atualizada com
sucesso.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
      format.json { render json: @operacao_com_vagao.errors, status: :unprocessable_entity }
    end
  end
end

def limparComposicao
  @operacao_com_vagao = OperacaoComVagao.find(params[:operacao_com_vagao][:id])

  respond_to do |format|
    if @operacao_com_vagao.update_attributes(:composicao_de_vagao_id => nil)
      @operacao_com_vagao.vt.update_attributes(:vt_carregado => true)
      format.html { redirect_to @operacao_com_vagao, notice: 'Operação com vagão atualizada com sucesso.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
      format.json { render json: @operacao_com_vagao.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /operacao_com_vagaos/1
# DELETE /operacao_com_vagaos/1.json
def destroy
  @operacao_com_vagao = OperacaoComVagao.find(params[:id])
  @operacao_com_vagao.vt.update_attributes(:vt_carregado => false)
  @operacao_com_vagao.destroy

  respond_to do |format|
    format.html { redirect_to operacao_com_vagaos_url }
    format.json { head :ok }
  end
end

end
```

```
# encoding: utf-8
class OperacaosController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :new]
  def liberarAcesso
```

```
  if @usuario.privilegio.id == 1
  return true
  else
    redirect_to "/static/grupo_diferente"
  return false
  end
end


# GET /operacaos
# GET /operacaos.json
def index
  @operacaos = Operacao.all

  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @operacaos }
  end
end

# GET /operacaos/1
# GET /operacaos/1.json
def show
  @operacao = Operacao.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @operacao }
  end
end

# GET /operacaos/new
# GET /operacaos/new.json
def new
  @operacao = Operacao.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @operacao }
  end
end

# GET /operacaos/1/edit
def edit
  @operacao = Operacao.find(params[:id])
end

# POST /operacaos
# POST /operacaos.json
def create
  @operacao = Operacao.new(params[:operacao])

  respond_to do |format|
    if @operacao.save
      format.html { redirect_to @operacao, notice: 'Operação criada com sucesso.' }
      format.json { render json: @operacao, status: :created, location: @operacao }
    else
      format.html { render action: "new" }
      format.json { render json: @operacao.errors, status: :unprocessable_entity }
    end
```

```
  end
 end

 # PUT /operacaos/1
 # PUT /operacaos/1.json
 def update
  @operacao = Operacao.find(params[:id])

  respond_to do |format|
   if @operacao.update_attributes(params[:operacao])
    format.html { redirect_to @operacao, notice: 'Operação atualizada com sucesso.' }
    format.json { head :ok }
   else
    format.html { render action: "edit" }
    format.json { render json: @operacao.errors, status: :unprocessable_entity }
   end
  end
 end

 # DELETE /operacaos/1
 # DELETE /operacaos/1.json
 def destroy
  @operacao = Operacao.find(params[:id])

  respond_to do |format|
   if @operacao.amostras.empty?()
    @operacao.destroy
    format.html { redirect_to operacaos_url }
    format.json { head :ok }
   else
    format.html { redirect_to @operacao, alert: ['ERRO: Não foi possível excluir a operação pois existem
amostras que utilizam ela.'] }
   end
  end
 end
end




# encoding: utf-8
class OrigemsController < ApplicationController

 before_filter :liberarAcesso, :except => [:index, :show, :new]
 def liberarAcesso
  if @usuario.privilegio.id == 1
  return true
  else
   redirect_to "/static/grupo_diferente"
  return false
  end
 end


 # GET /origems
 # GET /origems.json
 def index
  @origems = Origem.all
```

```
  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @origems }
  end
end

# GET /origems/1
# GET /origems/1.json
def show
  @origem = Origem.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @origem }
  end
end

# GET /origems/new
# GET /origems/new.json
def new
  @origem = Origem.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @origem }
  end
end

# GET /origems/1/edit
def edit
  @origem = Origem.find(params[:id])
end

# POST /origems
# POST /origems.json
def create
  @origem = Origem.new(params[:origem])

  respond_to do |format|
    if @origem.save
      format.html { redirect_to @origem, notice: 'Origem criada com sucesso.' }
      format.json { render json: @origem, status: :created, location: @origem }
    else
      format.html { render action: "new" }
      format.json { render json: @origem.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /origems/1
# PUT /origems/1.json
def update
  @origem = Origem.find(params[:id])

  respond_to do |format|
    if @origem.update_attributes(params[:origem])
      format.html { redirect_to @origem, notice: 'Origem atualizada com sucesso.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
```

```
      format.json { render json: @origem.errors, status: :unprocessable_entity }
    end
  end
 end

 # DELETE /origems/1
 # DELETE /origems/1.json
 def destroy
  @origem = Origem.find(params[:id])

  respond_to do |format|
   if @origem.amostras.empty?()
    @origem.destroy
    format.html { redirect_to origems_url }
    format.json { head :ok }
   else
    format.html { redirect_to @origem, alert: ['ERRO: Não foi possível excluir a origem pois existem amostras
que utilizam ela.'] }
   end
  end
 end
end




# encoding: UTF-8
class ProblemasController < ApplicationController

 before_filter :liberarAcesso, :except => [:index, :show, :new]
 def liberarAcesso
  if @usuario.privilegio.id == 1
  return true
  else
   redirect_to "/static/grupo_diferente"
  return false
  end
 end


 # GET /problemas
 # GET /problemas.json
 def index
  @problemas = Problema.all

  respond_to do |format|
   format.html # index.html.erb
   format.json { render json: @problemas }
  end
 end

 # GET /problemas/1
 # GET /problemas/1.json
 def show
  @problema = Problema.find(params[:id])

  respond_to do |format|
   format.html # show.html.erb
   format.json { render json: @problema }
```

```
  end
end

# GET /problemas/new
# GET /problemas/new.json
def new
  @problema = Problema.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @problema }
  end
end

# GET /problemas/1/edit
def edit
  @problema = Problema.find(params[:id])
end

# POST /problemas
# POST /problemas.json
def create
  @problema = Problema.new(params[:problema])

  respond_to do |format|
    if @problema.save
      format.html { redirect_to @problema, notice: 'Problema criado com sucesso.' }
      format.json { render json: @problema, status: :created, location: @problema }
    else
      format.html { render action: "new" }
      format.json { render json: @problema.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /problemas/1
# PUT /problemas/1.json
def update
  @problema = Problema.find(params[:id])

  respond_to do |format|
    if @problema.update_attributes(params[:problema])
      format.html { redirect_to @problema, notice: 'Problema atualizado com sucesso.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
      format.json { render json: @problema.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /problemas/1
# DELETE /problemas/1.json
def destroy
  @problema = Problema.find(params[:id])
  respond_to do |format|
    if @problema.vistorias.empty?()
      @problema.destroy
      format.html { redirect_to problemas_url }
      format.json { head :ok }
```

```
    else
     format.html { redirect_to @problema, alert: ['ERRO: Não foi possível excluir o problema
       pois existem vistorias que utilizam ele.'] }
    end
   end
  end
end




# encoding: utf-8
class ProdutosController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :new]
  def liberarAcesso
   if @usuario.privilegio.id == 1
   return true
   else
    redirect_to "/static/grupo_diferente"
   return false
   end
  end


  # GET /produtos
  # GET /produtos.json
  def index
   @produtos = Produto.all

   respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @produtos }
   end
  end

  # GET /produtos/1
  # GET /produtos/1.json
  def show
   @produto = Produto.find(params[:id])

   respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @produto }
   end
  end

  # GET /produtos/new
  # GET /produtos/new.json
  def new
   @produto = Produto.new

   respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @produto }
   end
  end

  # GET /produtos/1/edit
```

```
def edit
  @produto = Produto.find(params[:id])
end

# POST /produtos
# POST /produtos.json
def create
  @produto = Produto.new(params[:produto])

  respond_to do |format|
    if @produto.save
      format.html { redirect_to @produto, notice: 'Produto criado com sucesso.' }
      format.json { render json: @produto, status: :created, location: @produto }
    else
      format.html { render action: "new" }
      format.json { render json: @produto.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /produtos/1
# PUT /produtos/1.json
def update
  @produto = Produto.find(params[:id])

  respond_to do |format|
    if @produto.update_attributes(params[:produto])
      format.html { redirect_to @produto, notice: 'Produto atualizada com sucesso.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
      format.json { render json: @produto.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /produtos/1
# DELETE /produtos/1.json
def destroy
  @produto = Produto.find(params[:id])
  respond_to do |format|
    erros = []
    if @produto.amostras.empty?() and
      @produto.produto_vts.empty?()

      @produto.destroy
      format.html { redirect_to produtos_url }
      format.json { head :ok }
    end
    if !@produto.amostras.empty?()
      erros << "ERRO: Não foi possível excluir o produto pois existem amostras que utilizam ele."
    end
    if !@produto.produto_vts.empty?()
      erros << "ERRO: Não foi possível excluir o produto pois existem produtos de vagões que utilizam ele."
    end
    if erros != []
      format.html { redirect_to @produto, alert: erros }
    end
  end
end
```

```
end



# encoding: utf-8
class ProdutoVtsController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :new]
  def liberarAcesso
    if @usuario.privilegio.id == 2
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
  end


  # GET /produto_vts
  # GET /produto_vts.json
  def index
    @produto_vts = ProdutoVt.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @produto_vts }
    end
  end

  # GET /produto_vts/1
  # GET /produto_vts/1.json
  def show
    @produto_vt = ProdutoVt.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @produto_vt }
    end
  end

  # GET /produto_vts/new
  # GET /produto_vts/new.json
  def new
    @produto_vt = ProdutoVt.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @produto_vt }
    end
  end

  # GET /produto_vts/1/edit
  def edit
    @produto_vt = ProdutoVt.find(params[:id])
  end

  # POST /produto_vts
  # POST /produto_vts.json
```

```
  def create
    @produto_vt = ProdutoVt.new(params[:produto_vt])

    respond_to do |format|
      if @produto_vt.save
        format.html { redirect_to @produto_vt, notice: 'Produto de vagões criado com sucesso.' }
        format.json { render json: @produto_vt, status: :created, location: @produto_vt }
      else
        format.html { render action: "new" }
        format.json { render json: @produto_vt.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /produto_vts/1
  # PUT /produto_vts/1.json
  def update
    @produto_vt = ProdutoVt.find(params[:id])

    respond_to do |format|
      if @produto_vt.update_attributes(params[:produto_vt])
        format.html { redirect_to @produto_vt, notice: 'Produto de vagões atualizado com sucesso.' }
        format.json { head :ok }
      else
        format.html { render action: "edit" }
        format.json { render json: @produto_vt.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /produto_vts/1
  # DELETE /produto_vts/1.json
  def destroy
    @produto_vt = ProdutoVt.find(params[:id])
    respond_to do |format|
      if @produto_vt.operacao_com_vagaos.empty?()
        @produto_vt.destroy
        format.html { redirect_to produto_vts_url }
        format.json { head :ok }
      else
        format.html { redirect_to @produto_vt, alert: ['ERRO: Não foi possível excluir o produto de vagões
          pois existem operações com vagões que utilizam ele.']}
      end
    end
  end
end




# encoding: utf-8
class RelatoriosController < ApplicationController

  before_filter :liberarAcesso, :except =>  [
    :gerar_etiquetas,
    :gerar_relatorio_tempos_composicoes_vagoes,
    :gerar_relatorio_fechamento_diario_vagoes,
    :index
    ]
```

```
  def liberarAcesso
    if @usuario.privilegio.id == 2 or @usuario.privilegio.id == 3
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
  end

  def gerarRelatorioAmostras

    @amostras = []
    lista = []
    params.each do |p|
      lista = p
      if lista[1] == 'id'
        @amostras << Amostra.find(lista[0])
      end
    end
#render "relatorios/gerar_recibo_de_amostras"

    respond_to do |format|
      format.html {

        #html = render_to_string(:layout => 'relatorios')
        if params[:relatorio] == 'Gerar etiquetas'
          html = render_to_string(:template => "relatorios/gerar_etiquetas")
        elsif params[:relatorio] == 'Gerar recibo de amostras'
          html = render_to_string(:template => "relatorios/gerar_recibo_de_amostras")
        end

        kit = PDFKit.new(html, :page_size => "A4")
        send_data(kit.to_pdf, :filename => "etiquetas.pdf", :type => 'application/pdf')
        return
      }
    end

  end

  def gerar_relatorio_tempos_composicoes_vagoes
    @composicao_de_vagaos = ComposicaoDeVagao.last(500).sort_by{|composicao_de_vagao|
[composicao_de_vagao.entrada_plataforma]}.reverse

    respond_to do |format|
      format.html {
       #@composicao_de_vagaos
        html = render_to_string(:layout => 'relatorios')
        kit = PDFKit.new(html, :orientation => :landscape, :page_size => :A4)
        #kit.stylesheets << "#{Rails.root}/public/stylesheets/screen.css"
        send_data(kit.to_pdf, :filename => "relatorio tempos composicoes vagoes.pdf", :type => 'application/pdf')
        return # to avoid double render call
      }
    end

  end

  def gerar_relatorio_fechamento_diario_vagoes

    @fechamento_diario_vagao = FechamentoDiarioVagao.first
```

```
  respond_to do |format|
   format.html {
    #@fechamento_diario_vagaos
      html = render_to_string(:layout => 'relatorios')
      kit = PDFKit.new(html, :page_size => "A4")
      #kit.stylesheets << "#{Rails.root}/public/stylesheets/screen.css"
      send_data(kit.to_pdf, :filename => "relatorio fechamento diario vagoes.pdf", :type => 'application/pdf')
      return # to avoid double render call
    }
  end

 end

 def index
  render :layout => 'application', :template => 'relatorios/index'
 end

end




#encoding: utf-8

class SolicitacaosController < ApplicationController

 before_filter :liberarAcesso, :except => [:index, :show, :new]
 def liberarAcesso
  if @usuario.privilegio.id == 2
  return true
  else
    redirect_to "/static/grupo_diferente"
  return false
  end
 end


 # GET /solicitacaos
 # GET /solicitacaos.json
 def index
  @solicitacaos = Solicitacao.all

  respond_to do |format|
    format.html # index.html.erb
    format.json { render json: @solicitacaos }
  end
 end

 # GET /solicitacaos/1
 # GET /solicitacaos/1.json
 def show
  @solicitacao = Solicitacao.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @solicitacao }
  end
 end
```

```
  # GET /solicitacaos/new
  # GET /solicitacaos/new.json
  def new
    @solicitacao = Solicitacao.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @solicitacao }
    end
  end

  # GET /solicitacaos/1/edit
  def edit
    @solicitacao = Solicitacao.find(params[:id])
  end

  # POST /solicitacaos
  # POST /solicitacaos.json
  def create
    @solicitacao = Solicitacao.new(params[:solicitacao])

    respond_to do |format|
      if @solicitacao.save
        format.html { redirect_to @solicitacao, notice: 'Solicitação criada com sucesso.' }
        format.json { render json: @solicitacao, status: :created, location: @solicitacao }
      else
        format.html { render action: "new" }
        format.json { render json: @solicitacao.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /solicitacaos/1
  # PUT /solicitacaos/1.json
  def update
    @solicitacao = Solicitacao.find(params[:id])

    respond_to do |format|
      if @solicitacao.update_attributes(params[:solicitacao])
        format.html { redirect_to @solicitacao, notice: 'Solicitação atualizada com sucesso.' }
        format.json { head :ok }
      else
        format.html { render action: "edit" }
        format.json { render json: @solicitacao.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /solicitacaos/1
  # DELETE /solicitacaos/1.json
  def destroy
    @solicitacao = Solicitacao.find(params[:id])
    @solicitacao.destroy

    respond_to do |format|
      format.html { redirect_to solicitacaos_url }
      format.json { head :ok }
    end
  end
end
```

```
class StaticController < ApplicationController
  def static
    if params[:id] && template_exists?(params[:id], ["static"])
      render params[:id]
    else
      redirect_to '/404.html', :status => 404
    end
  end

end
```

```
# encoding: utf-8
class TipoAmostrasController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :new]
  def liberarAcesso
    if @usuario.privilegio.id == 1
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
  end


  # GET /tipo_amostras
  # GET /tipo_amostras.json
  def index
    @tipo_amostras = TipoAmostra.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @tipo_amostras }
    end
  end

  # GET /tipo_amostras/1
  # GET /tipo_amostras/1.json
  def show
    @tipo_amostra = TipoAmostra.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @tipo_amostra }
    end
  end

  # GET /tipo_amostras/new
  # GET /tipo_amostras/new.json
  def new
    @tipo_amostra = TipoAmostra.new
```

```
  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @tipo_amostra }
  end
 end

 # GET /tipo_amostras/1/edit
 def edit
  @tipo_amostra = TipoAmostra.find(params[:id])
 end

 # POST /tipo_amostras
 # POST /tipo_amostras.json
 def create
  @tipo_amostra = TipoAmostra.new(params[:tipo_amostra])

  respond_to do |format|
    if @tipo_amostra.save
      format.html { redirect_to @tipo_amostra, notice: 'Tipo de amostra criada com sucesso.' }
      format.json { render json: @tipo_amostra, status: :created, location: @tipo_amostra }
    else
      format.html { render action: "new" }
      format.json { render json: @tipo_amostra.errors, status: :unprocessable_entity }
    end
  end
 end

 # PUT /tipo_amostras/1
 # PUT /tipo_amostras/1.json
 def update
  @tipo_amostra = TipoAmostra.find(params[:id])

  respond_to do |format|
    if @tipo_amostra.update_attributes(params[:tipo_amostra])
      format.html { redirect_to @tipo_amostra, notice: 'Tipo de amostra atualizada com sucesso.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
      format.json { render json: @tipo_amostra.errors, status: :unprocessable_entity }
    end
  end
 end

 # DELETE /tipo_amostras/1
 # DELETE /tipo_amostras/1.json
 def destroy
  @tipo_amostra = TipoAmostra.find(params[:id])
  respond_to do |format|
    if @tipo_amostra.amostras.empty?()
      @tipo_amostra.destroy
      format.html { redirect_to tipo_amostras_url }
      format.json { head :ok }
    else
      format.html { redirect_to @tipo_amostra, alert: ['ERRO: Não foi possível excluir o tipo de amostra pois
existem amostras que utilizam ele.'] }
    end
  end
 end
end
```

```
# encoding: UTF-8
class UsuariosController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :edit, :update, :new]
  def liberarAcesso
    if @usuario.privilegio.id == 1
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
  end

  # GET /usuarios
  # GET /usuarios.json
  def index
    @usuarios = Usuario.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @usuarios }
    end
  end

  # GET /usuarios/1
  # GET /usuarios/1.json
  def show
    @usuario = Usuario.find(params[:id])
    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @usuario }
    end
  end

  # GET /usuarios/new
  # GET /usuarios/new.json
  def new
    @usuario = Usuario.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @usuario }
    end
  end

  # GET /usuarios/1/edit
  def edit
    @usuario = Usuario.find(params[:id])
  end

  # POST /usuarios
  # POST /usuarios.json
  def create
    @usuario = Usuario.new(params[:usuario])
    @usuario.chave.upcase!
```

```
    @usuario.senha = Digest::SHA2.hexdigest("linux #{@usuario.senha} sudo su").force_encoding('UTF-8')

    respond_to do |format|
      if @usuario.save
        format.html { redirect_to @usuario, notice: 'Usuário criado com sucesso.' }
        format.json { render json: @usuario, status: :created, location: @usuario }
      else
        format.html { render action: "new" }
        format.json { render json: @usuario.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /usuarios/1
  # PUT /usuarios/1.json
  def update
    @usuario_logado = @usuario
    @usuario = Usuario.find(params[:id])
    respond_to do |format|
      params[:usuario][:chave].upcase!
      if  @usuario_logado.id == @usuario.id

        if @usuario.update_attribute(:senha, Digest::SHA2.hexdigest("linux #{params[:usuario][:senha]} sudo
su").force_encoding('UTF-8'))

          format.html { redirect_to @usuario, notice: 'Usuário atualizado com sucesso.' }
          format.json { head :ok }

        else

          format.html { render action: "edit" }
          format.json { render json: @usuario.errors, status: :unprocessable_entity }

        end

      elsif @usuario_logado.privilegio.id == 1
        if @usuario.id != 1

          if @usuario.update_attributes(params[:usuario]) and
          @usuario.update_attribute(:senha, Digest::SHA2.hexdigest("linux #{@usuario.senha} sudo
su").force_encoding('UTF-8'))

            format.html { redirect_to @usuario, notice: 'Usuário atualizado com sucesso.' }
            format.json { head :ok }

          else
            format.html { render action: "edit" }
            format.json { render json: @usuario.errors, status: :unprocessable_entity }
          end
        end

      else
        format.html { render action: "edit" }
        format.json { render json: @usuario.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /usuarios/1
  # DELETE /usuarios/1.json
```

```
  def destroy
    @usuario = Usuario.find(params[:id])
    respond_to do |format|
      erros = []
      if !@usuario.amostras.empty?()
        erros << "ERRO: Não foi possível excluir o usuário pois existem amostras que pertencem a ele."
      end
      if !@usuario.composicao_de_vagaos.empty?()
        erros << "ERRO: Não foi possível excluir o usuário pois existem composições de vagões que pertencem a
ele."
      end
      if !@usuario.nota_vts.empty?()
        erros << "ERRO: Não foi possível excluir o usuário pois existem notas de vagões que pertencem a ele."
      end
      if @usuario.id == 1
        erros << "OPA! Isso não é legal. Você não pode excluir o Super Usuário."
      end
      if erros == []
        if @usuario.destroy
          format.html { redirect_to usuarios_url }
          format.json { head :ok }
        end
      else
        format.html { redirect_to @usuario, alert: erros }
      end
    end
  end
end




class VistoriasController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :new]
  def liberarAcesso
    if @usuario.privilegio.id == 2
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
  end


  # GET /vistorias
  # GET /vistorias.json
  def index
    @vistorias = Vistoria.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @vistorias }
    end
  end

  # GET /vistorias/1
  # GET /vistorias/1.json
  def show
```

```
  @vistoria = Vistoria.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @vistoria }
  end
end

# GET /vistorias/new
# GET /vistorias/new.json
def new
  @vistoria = Vistoria.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @vistoria }
  end
end

# GET /vistorias/1/edit
def edit
  @vistoria = Vistoria.find(params[:id])
end

# POST /vistorias
# POST /vistorias.json
def create
  @vistoria = Vistoria.new(params[:vistoria])

  respond_to do |format|
    if @vistoria.save
      format.html { redirect_to @vistoria, notice: 'Vistoria criada com sucesso.' }
      format.json { render json: @vistoria, status: :created, location: @vistoria }
    else
      format.html { render action: "new" }
      format.json { render json: @vistoria.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /vistorias/1
# PUT /vistorias/1.json
def update
  @vistoria = Vistoria.find(params[:id])

  respond_to do |format|
    if @vistoria.update_attributes(params[:vistoria])
      format.html { redirect_to @vistoria, notice: 'Vistoria atualizada com sucesso.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
      format.json { render json: @vistoria.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /vistorias/1
# DELETE /vistorias/1.json
def destroy
  @vistoria = Vistoria.find(params[:id])
```

```
    @vistoria.destroy

    respond_to do |format|
      format.html { redirect_to vistorias_url }
      format.json { head :ok }
    end
  end
end




# encoding: UTF-8
class VtsController < ApplicationController

  before_filter :liberarAcesso, :except => [:index, :show, :menu_vt, :new]
  def liberarAcesso
    if @usuario.privilegio.id == 1
    return true
    else
      redirect_to "/static/grupo_diferente"
    return false
    end
  end

  def menu_vt
    render :layout => 'application', :template => 'vts/menu_vt'
  end

  # GET /vts
  # GET /vts.json
  def index
    @vts = Vt.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @vts }
    end
  end

  # GET /vts/1
  # GET /vts/1.json
  def show
    @vt = Vt.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @vt }
    end
  end

  # GET /vts/new
  # GET /vts/new.json
  def new
    @vt = Vt.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @vt }
```

```
  end
end

# GET /vts/1/edit
def edit
  @vt = Vt.find(params[:id])
end

# POST /vts
# POST /vts.json
def create
  @vt = Vt.new(params[:vt])

  respond_to do |format|
    if @vt.save
      format.html { redirect_to @vt, notice: 'Vagão criado com sucesso.' }
      format.json { render json: @vt, status: :created, location: @vt }
    else
      format.html { render action: "new" }
      format.json { render json: @vt.errors, status: :unprocessable_entity }
    end
  end
end

# PUT /vts/1
# PUT /vts/1.json
def update
  @vt = Vt.find(params[:id])

  respond_to do |format|
    if @vt.update_attributes(params[:vt])
      format.html { redirect_to @vt, notice: 'Vagão atualizado com sucesso.' }
      format.json { head :ok }
    else
      format.html { render action: "edit" }
      format.json { render json: @vt.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /vts/1
# DELETE /vts/1.json
def destroy
  @vt = Vt.find(params[:id])
  respond_to do |format|
    erros = []

    if @vt.vistorias.empty?() and
    @vt.operacao_com_vagaos.empty?()

      @vt.destroy
      format.html { redirect_to vts_url }
      format.json { head :ok }
    end

    if !@vt.vistorias.empty?()
      erros << 'ERRO: Não foi possível excluir o vagão tanque
        pois existem vistorias que pertencem a ele.'
    end
```

```
    if !@vt.operacao_com_vagaos.empty?()
      erros << 'ERRO: Não foi possível excluir o vagão tanque
        pois existem operações com vagões que utilizam ele.'
    end

    if erros != []
      format.html { redirect_to @vt, alert: erros }
    end

  end
 end
end
```

9.8 CÓDIGO FONTE DOS MODELOS

```
# encoding: UTF-8
class Amostra < ActiveRecord::Base

  #validates :data                ,:inclusion => { :in => 30000..50000 } ,validates_uniqueness_of :name,
:case_sensitive => false, :presence =>  true, :in => 6..20, :with => /\A[a-zA-Z]+\z/, :length => { :within =>
1..10000000 }, :length => { :maximum => 100 }  , :uniqueness =>  true, :uniqueness => { :case_sensitive =>
false } , :numericality =>  true , :inclusion  =>  { :in =>  0..9  } , :format =>  { :with =>  %r\A([^@\s]
+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\Z/,  :on =>  :create }
  validates :certificado_Ilab      , :uniqueness => { :case_sensitive => false }, :allow_blank => true
  validates :numero_gerado_Labware , :uniqueness =>  true, :numericality =>  true, :allow_blank => true
#, :inclusion  =>  { :in =>  0..9  }
  validates_presence_of :responsavel, :operacao, :origem, :tipo_amostra, :produto

  validate :validarDataLimite


  belongs_to :responsavel, :class_name => 'Usuario'
  belongs_to :operacao
  belongs_to :origem
  belongs_to :tipo_amostra
  belongs_to :produto

  has_many :lacres, :dependent => :destroy

  @@imprimir_etiqueta = false
  def isImprimirEtiqueta
    @@imprimir_etiqueta
  end

  def imprimirEtiqueta
    @@imprimirEtiqueta = true
  end

  def amostraEncontrada
    r = ""
    if numero_gerado_Labware != nil
      r = "sim"
    elsif (created_at > (Date.today - 30)) and (numero_gerado_Labware == nil)
      r = "não encontrada atual"
```

```
    elsif (created_at < (Date.today - 30)) and (numero_gerado_Labware == nil)
      r = "não encontrada antiga"
    end
    r
  end

  def pertenceAUmGrupo?(amostras)
    amostras = amostras.sort_by{|amostra| [amostra.id]}.reverse
    retorno = false

    if responsavel == amostras[0].responsavel and
      operacao == amostras[0].operacao and
      origem == amostras[0].origem and
      tipo_amostra == amostras[0].tipo_amostra and
      produto == amostras[0].produto and
      75 > (amostras[0].created_at - created_at)
      retorno = true
    end

    retorno
  end

  private

  def validarDataLimite
    if self.created_at != nil
      if self.created_at < (Date.today - 30)
        errors.add :id, ": Amostra criada a mais de um mês não pode ser alterada."
      end
    end
  end

  public

end




#encoding: utf-8
class AmostrasNavio
  include ActiveModel::Validations
  include ActiveModel::Conversion
  extend ActiveModel::Naming

  attr_accessor :amostras, :data, :responsavel, :operacao, :produto, :nome_navio, :viagem, :berco, :tanques,
:observacao, :finalidades, :lacres

  validates_presence_of :nome_navio, :viagem, :berco, :tanques, :finalidades
  validate :validarAmostra
  validate :validarLacres
  def initialize(attributes = {:tanques => [], :finalidades => []})
    attributes.each do |name, value|
      send("#{name}=", value)
    end
  end

  def persisted?
    false
```

```
  end

  def criarAmostraNavio
    amostra = Amostra.new(:data => data,
    :responsavel_id => responsavel,
    :operacao_id => operacao,
    :produto_id => produto,
    :origem_id => 44,
    :tipo_amostra_id => 1)
    amostra
  end

  def grupoAmostras

    amostras = []
    if tanques == []
      amostra =  criarAmostraNavio
      amostra.observacao = "Nome do navio / Vessel: #{nome_navio}. Viagem / Voyage: #{viagem}. Berço /
Berth: #{berco}. Tanque / Tank: #{tanques}. Obs.: #{observacao}"
    amostras << amostra
    else
      tanques.each do |tq|
        amostra = criarAmostraNavio
        amostra.observacao = "Nome do navio / Vessel: #{nome_navio}. Viagem / Voyage: #{viagem}. Berço /
Berth: #{berco}. Tanque / Tank: #{tq}. Obs.: #{observacao}"
        amostras << amostra
      end
    end

    amostras
  end

  def numeroEtiquetasComLacre
    if finalidades.include?("analise_terminal")
      etiquetas_com_lacre = tanques.length * (finalidades.length - 1)
    else
      etiquetas_com_lacre = tanques.length * finalidades.length
    end
    etiquetas_com_lacre
  end

  private

  def validarAmostra
    if self.tanques == nil
      self.tanques = []
    end
    if self.finalidades == nil
      self.finalidades = []
    end
    amostra = criarAmostraNavio
    if !amostra.valid?
      amostra.errors.full_messages.each do |msg|
        errors.add :amostras, ": #{msg}"
      end
    end
    if ![1,2,10].include?(operacao.to_i)
      errors.add :operacao, ": Operação inválida para este tipo de amostra"
    end
```

```
  end

  def validarLacres
    if lacres.split.length != 0

      lacres_array = lacres.split

      if lacres_array.length != numeroEtiquetasComLacre
        errors.add :lacres, ": Você está tentando incluir #{lacres_array.length} lacre(s), quando o número
requerido é #{numeroEtiquetasComLacre}"
      end
      lacre_teste = Lacre.new(:amostra_id => Amostra.last.id, :finalidade_id => Finalidade.last.id)
      lacres_array.each do |l|
        lacre_teste.numero = l
        if !lacre_teste.valid?
          lacre_teste.errors.full_messages.each do |msg|
            errors.add :lacres, ": #{l} #{msg}"
          end
        end
      end

      if lacres_array.uniq.length != lacres_array.length
        errors.add :lacres, ": Você está tentando incluir lacres com números repetidos"
      end
    end
  end

  public

end




# encoding: utf-8
class ComposicaoDeVagao < ActiveRecord::Base

  validates_presence_of :responsavel, :fechamento_diario_vagao
  validates :temperatura, :numericality =>  true, :inclusion => { :in => 10..90, :message => "fora dos
limites" }, :allow_blank => true
  validates :tanque_destino, :format => {:with => /^[0-9]{3,3}$/}, :allow_blank => true
  validate :validarDatas
  validate :validarPossibilidadeDeAlteracao
  validate :validarOperacaoComVagaos

  belongs_to :responsavel, :class_name => "Usuario"
  belongs_to :fechamento_diario_vagao

  has_many :solicitacaos, :dependent => :destroy
  has_many :operacao_com_vagaos

  attr_accessor :conjunto_operacoes_de_vagoes

  private
  def validarOperacaoComVagaos
    if operacao_com_vagaos != nil
      operacao_com_vagaos.each do |op|
        if !op.valid?
          op.errors.full_messages.each do |msg|
```

```
        errors.add :id, ": #{msg}"
       end
      end
    end
  end
end

def validarPossibilidadeDeAlteracao
  if operacoesFinalizadas?
    errors.add :id, ": Operação já está finalizada"
  end
end

def validarDatas
  if (self.inicio_descarga != nil and
  self.entrada_plataforma != nil and
  self.entrada_plataforma > self.inicio_descarga) or
  (self.inicio_descarga != nil and
  self.entrada_plataforma == nil)

    self.errors.add :inicio_descarga, ": Início da descarga deve ser após entrada na plataforma"
  end
  if (self.final_descarga != nil and
  self.inicio_descarga != nil and
  self.inicio_descarga > self.final_descarga) or
  (self.final_descarga != nil and
  self.inicio_descarga == nil)

    self.errors.add :final_descarga, ": Final da descarga deve ser após início da descarga"
  end
  if (self.saida_plataforma != nil and
  self.final_descarga != nil and
  self.final_descarga > self.saida_plataforma) or
  (self.saida_plataforma != nil and
  self.final_descarga == nil)

    self.errors.add :saida_plataforma, ": Saída da plataforma deve ser após final da descarga"
  end
  if self.id != nil and self.final_descarga == nil
    if ComposicaoDeVagao.find(self.id).final_descarga != nil
      self.errors.add :final_descarga, ": Final da descarga não pode ser nula"
    end
  end
end

public

def idEntradaPlataformaInicioDescarga

  s = "#{self.id} | Entrada: "
  if self.entrada_plataforma != nil
    s << self.entrada_plataforma.strftime("%d/%m/%Y %H:%M")
  end
  s << " | Início: "
  if self.inicio_descarga != nil
    s << self.inicio_descarga.strftime("%d/%m/%Y %H:%M")
  end
  s

end
```

```
def totalPeso
 peso = 0
 operacao_com_vagaos.each do |ov|
  peso = peso + ov.peso
 end
 peso
end

def qtdOperacoes
 operacao_com_vagaos.length
end

def produto
 produtos = []
 string = ''
 operacao_com_vagaos.each do |op|
  if !produtos.include?(op.produto_vt.descricao_produto.nome_curto)
   produtos << op.produto_vt.descricao_produto.nome_curto
   string = "#{string} |#{op.produto_vt.descricao_produto.nome_curto}|"
  end
 end
 string
end

def tempoParaInicio
 tempo =  inicio_descarga - entrada_plataforma
 tempo
end

def tempoParaLiberar
 tempo =  final_descarga - inicio_descarga
 tempo
end

def datasDasNotas
 datas = []
 operacao_com_vagaos.each do |op|
  if !datas.include?(op.nota_vt.data)
  datas << op.nota_vt.data
  end
 end
 datas
end

def tempoTranspetro
 tempo = tempoParaInicio + tempoParaLiberar
 tempo
end

def tempoAll
 comp_anterior = ComposicaoDeVagao.last(:conditions => ["final_descarga < ?", final_descarga])
 if comp_anterior != nil
  tempo = entrada_plataforma - comp_anterior.final_descarga
 else
 tempo = 0
 end
 tempo
end
```

```
  def consistente?
    consistente = true
    if entrada_plataforma == nil or
    inicio_descarga == nil or
    final_descarga == nil or
    saida_plataforma == nil or
    temperatura == nil or
    tanque_destino == nil or
    operacao_com_vagaos == []

    consistente = false

    end
    consistente
  end

  def operacoesFinalizadas?
    operacoes_finalizadas = true
    if operacao_com_vagaos == []
    operacoes_finalizadas = false
    end
    operacao_com_vagaos.each do |op|
      if !op.operacao_finalizada
      operacoes_finalizadas = false
      break
      end
    end
    operacoes_finalizadas
  end

end




#encoding: utf-8
class ConjuntoOperacoesDeVagoes
  include ActiveModel::Validations
  include ActiveModel::Conversion
  extend ActiveModel::Naming

  attr_accessor :vagoes, :pesos, :produto, :produto_para_confirmar, :nota_vt, :vts, :op_vts,
:composicao_de_vagao

  validates_presence_of :vagoes, :pesos, :produto

  validate :validarVagoesPesos

  validate :validarProdutoConfirmado

  validate :validarOpVagoes

  validate :validarOpVagoesAntesDeAtualizarOperacoes

  def initialize(attributes = {})
    attributes.each do |name, value|
      send("#{name}=", value)
    end
  end
```

```
  def persisted?
    false
  end

  def total
    total = 0
    pesos.split.each do |p|
      if true
      total = total + p.to_i
      end
    end
    total
  end

  def numeroDeOperacoes
    op_vts.length
  end

  def produtosDiferentes?
    diferentes = false
    produto = nil
    op_vts.each do |op|
      if produto == nil
        produto = op.produto_vt
      end
      if produto != op.produto_vt
        diferentes = true
      end
    end
    diferentes
  end

  private

  def validarVagoesPesos
    validarVagoes = true
    if vagoes.split.length != pesos.split.length and self.composicao_de_vagao == nil
      errors.add :vagoes, ": Os números dos vagões (#{vagoes.split.length}), não coincidem com os números dos
pesos (#{pesos.split.length})."
    validarOperacoes = false
    end
    if validarVagoes
      self.vts = []
      vagoes.split.each do |vagao|
        vt = Vt.where("numero_curto = ?", "#{(vagao.delete "-")[0..3]}-#{(vagao.delete "-")[4..
(vagao.length)]}").first
        if vt == nil
          errors.add :vagoes, ": Vagão (#{(vagao.delete "-")[0..3]}-#{(vagao.delete "-")[4..(vagao.length)]}) não
existe."
        else
        self.vts << vt
        end
      end
    end
  end

  def validarOpVagoes
    cont = 0
    p = pesos.split
```

```
    self.op_vts = []
    if vts != nil
      if vts.length == p.length and composicao_de_vagao == nil
        vts.each do |vt|
          opVt = OperacaoComVagao.new(:peso => p[cont], :vt_id => vt.id, :nota_vt_id => nota_vt.id,
:produto_vt_id => produto)
          cont = cont + 1
          if opVt.valid?
          self.op_vts << opVt
          else
            opVt.errors.full_messages.each do |msg|
              errors.add :vagoes, ": Vagão (#{vt.numero_curto}) - #{msg}"
            end
          end
        end
      end
    end
  end

  def validarProdutoConfirmado
    if produto != produto_para_confirmar
      errors.add :produto, ": Confira o produto da nota"
    end
  end

  def validarOpVagoesAntesDeAtualizarOperacoes
    if composicao_de_vagao != nil
      vts.each do |vt|
        opVt = vt.operacao_com_vagaos.last
        if opVt.composicao_de_vagao == nil
          opVt.composicao_de_vagao = composicao_de_vagao
        self.op_vts << opVt
        else
          errors.add :vagoes, ": Operação com vagão (#{vt.numero_curto}) já pertence a composição
#{opVt.composicao_de_vagao.id}"
        end
        if vt.vt_carregado == false
          errors.add :vagoes, ": Vagão (#{vt.numero_curto}) está vazio"
        end
        if !opVt.valid?
          opVt.errors.full_messages.each do |msg|
            errors.add :vagoes, ": Vagão (#{vt.numero_curto}) - #{msg}"
          end
        end
      end
    end
  end

  public

end




#encoding: utf-8
class FechamentoDiarioVagao < ActiveRecord::Base

  validates :data, :uniqueness =>  true, :presence => true
```

```
validate :validarSeUltimaParaAlterar
validate :validarPossibilidadeDeAlteracao

has_many :composicao_de_vagaos

def pesoTotal
  p = 0
  composicao_de_vagaos.each do |cv|
    p = p + cv.totalPeso
  end
  p
end

def qtdOperacoes
  qtd = 0
  composicao_de_vagaos.each do |comp|
    qtd = qtd + comp.operacao_com_vagaos.length
  end
  qtd
end

def produto
  produtos = []
  string = ''
  composicao_de_vagaos.each do |comp|
    if !produtos.include?(comp.produto)
      produtos << comp.produto
      string = "#{string} #{comp.produto}"
    end
  end
  string
end

def composicoesFinalizado?
  composicoes_finalizadas = true
  if composicao_de_vagaos == []
    composicoes_finalizadas = false
  end
  composicao_de_vagaos.each do |comp|
    if !comp.operacoesFinalizadas?
    composicoes_finalizadas = false
    break
    end
  end
  composicoes_finalizadas
end

private

def validarSeUltimaParaAlterar
  if id != nil
    valido = false
    fechamentos = FechamentoDiarioVagao.last(2)
    fechamentos.each do |f|
      if f.id == id
        valido = true
      end
    end
    if !valido
      errors.add :id, ": Só é permitido alterar os dois últimos fechamentos de vagões."
```

```
    end
  end
end

def validarPossibilidadeDeAlteracao
  if composicoesFinalizado?
    errors.add :id, ": Operação já está finalizada"
  end
end

public
end




class Finalidade < ActiveRecord::Base

  validates :descricao, :uniqueness => { :case_sensitive => false }, :presence => true

  has_many :lacres
end




class GrupoUsuario < ActiveRecord::Base

  validates :tipo, :uniqueness => { :case_sensitive => false }, :presence => true
  validates :descricao, :uniqueness => { :case_sensitive => false }, :presence => true

  has_many :usuarios, :foreign_key => :privilegio_id #, :dependent => :destroy
end




class Lacre < ActiveRecord::Base

  validates_numericality_of :numero, :only_integer => true, :allow_blank => true, :greater_than => 0
  validates :numero, :uniqueness => true, :allow_blank => true
  validates_presence_of :amostra, :finalidade

  belongs_to :amostra
  belongs_to :finalidade
end




# encoding: UTF-8
class NotaVt < ActiveRecord::Base

  validates_numericality_of :numero_v2, :only_integer => true
  validates :numero_v2, :uniqueness => true, :presence => true,
```

```
        :inclusion => { :in => 2000..50000, :message => "é o número da V2 (V2 Nº) e não o número do
controle do formulário" }
  validates :data, :presence => true
  validates :responsavel, :presence => true

  belongs_to :responsavel, :class_name => 'Usuario'

  has_many :operacao_com_vagaos

  attr_accessor :conjunto_operacoes_de_vagoes

  def produtos
    produtos = []
    string = ''
    operacao_com_vagaos.each do |op|
      if !produtos.include?(op.produto_vt.descricao_produto.nome_curto)
        produtos << op.produto_vt.descricao_produto.nome_curto
        string = "#{string} |#{op.produto_vt.descricao_produto.nome_curto}|"
      end
    end
    string
  end

  def qtdOperacoes
    operacao_com_vagaos.length
  end

  def pesoTotal
    peso = 0
    operacao_com_vagaos.each do |ov|
      peso = peso + ov.peso
    end
    peso
  end

  def operacoesFinalizadas
    operacoes_finalizadas = true
    if operacao_com_vagaos == []
      operacoes_finalizadas = false
    end
    operacao_com_vagaos.each do |op|
      if !op.operacao_finalizada
      operacoes_finalizadas = false
      break
      end
    end
    operacoes_finalizadas
  end

end




class Operacao < ActiveRecord::Base

  validates :descricao, :uniqueness => { :case_sensitive => false }, :presence => true

  has_many :amostras
```

```
end



#encoding: utf-8
class OperacaoComVagao < ActiveRecord::Base

  validates :peso, :presence => true, :numericality =>  true, :inclusion => { :in => 30000..43100, :message =>
"fora dos limites (30.000 até 43.100)" }
  validates_presence_of :vt, :nota_vt, :produto_vt
  validate :validarVtVazioAoCriarNovo
  validate :validarPossibilidadeDeAlteracao
  validate :validarComposicaoVagao

  belongs_to :vt
  belongs_to :nota_vt
  belongs_to :produto_vt
  belongs_to :composicao_de_vagao
  def numeroVagao
    vt.numero_curto
  end

  private

  def validarVtVazioAoCriarNovo
    if self.vt != nil
      if self.vt.vt_carregado and self.id == nil
        errors.add :vt, "já está carregado"
      end
    end
  end

  def validarPossibilidadeDeAlteracao
    if id != nil
      if OperacaoComVagao.find(self.id).operacao_finalizada
        errors.add :operacao_finalizada, ": Operação já está finalizada"
      end
    end
  end

  def validarComposicaoVagao
    if composicao_de_vagao != nil
      if composicao_de_vagao.operacoesFinalizadas?
        errors.add :composicao_de_vagao, ": Composição de vagão já está finalizada"
      end
    end
  end

end



class Origem < ActiveRecord::Base

  validates :descricao, :uniqueness => { :case_sensitive => false }, :presence => true
```

```
  has_many :amostras
end



class Problema < ActiveRecord::Base

  validates :descricao, :uniqueness => { :case_sensitive => false }, :presence => true

  has_and_belongs_to_many :vistorias

end



class Produto < ActiveRecord::Base

  validates :nome_completo, :uniqueness => { :case_sensitive => false }, :presence => true
  validates :nome_curto, :uniqueness => { :case_sensitive => false }, :presence => true
  validates :codigo, :uniqueness => { :case_sensitive => false }, :presence => true

  has_many :amostras
  has_many :produto_vts, :foreign_key => :descricao_produto_id


  def descricaoCompleta
    descricao = "#{self.codigo} | "
    descricao << "#{self.nome_curto} | "
    descricao << self.nome_completo
    descricao
  end

  def temBenzeno?
    if [1,2,3,34,35,36].include?(id)
      true
    else
      false
    end
  end

end



class ProdutoVt < ActiveRecord::Base

  validates :certificado, :uniqueness => { :case_sensitive => false }, :presence => true
  validates_presence_of :descricao_produto

  belongs_to :descricao_produto, :class_name => 'Produto'
  has_many :operacao_com_vagaos

  def nomeProduto
    descricao = descricao_produto.nome_curto
```

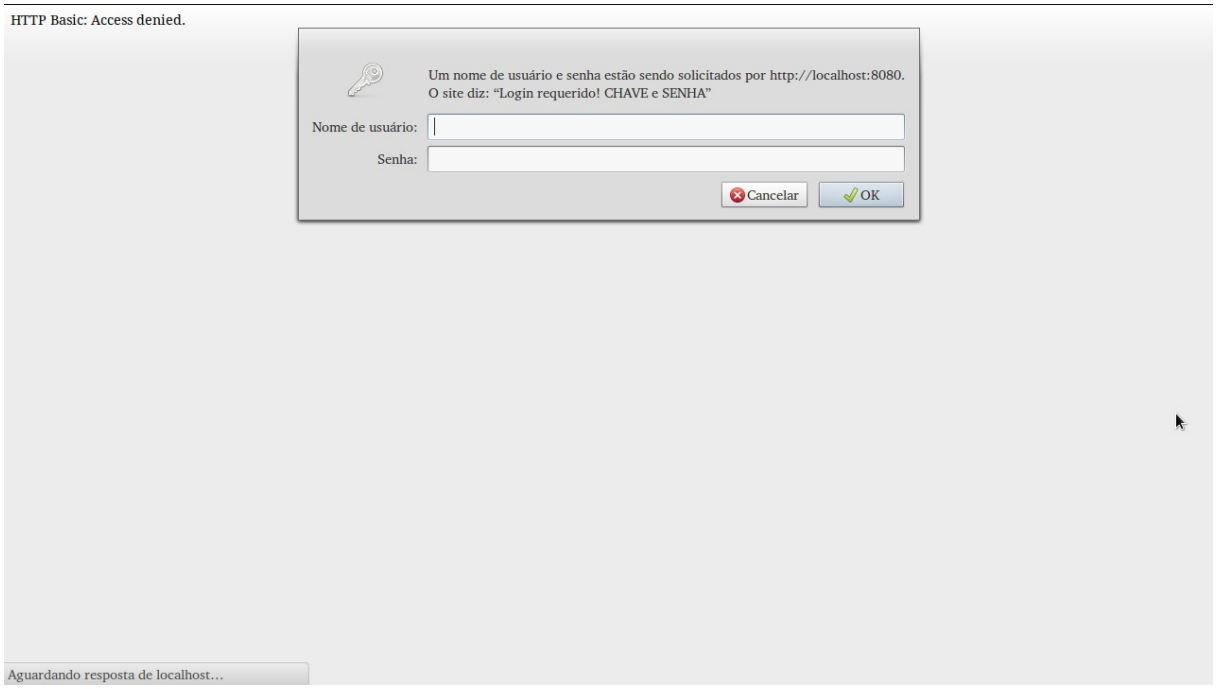
```
    descricao
  end

  def produtoECertificado
    pc = "#{descricao_produto.nome_curto} | (CE - #{certificado})"
    pc
  end
end




class Solicitacao < ActiveRecord::Base

  validates :data_hora, :uniqueness => true, :presence => true
  validates :responsavel_atender_solicitacao, :presence => true
  validates_presence_of :composicao_de_vagao, :descricao

  belongs_to :composicao_de_vagao
end




class TipoAmostra < ActiveRecord::Base

  validates :descricao, :uniqueness => { :case_sensitive => false }, :presence => true

  has_many :amostras
end




class Usuario < ActiveRecord::Base

  validates :chave, :uniqueness => { :case_sensitive => false }, :presence => true,
          :format => {:with => /^[A-Z0-9]{4,4}$/}
  validates :nome, :presence => true
  validates :nome_guerra, :presence => true
  validates_presence_of :privilegio

  belongs_to :privilegio, :class_name => 'GrupoUsuario'

  has_many :amostras, :foreign_key => :responsavel_id
  has_many :composicao_de_vagaos, :foreign_key => :responsavel_id
  has_many :nota_vts, :foreign_key => :responsavel_id

end




# encoding: utf-8
class Vistoria < ActiveRecord::Base
```

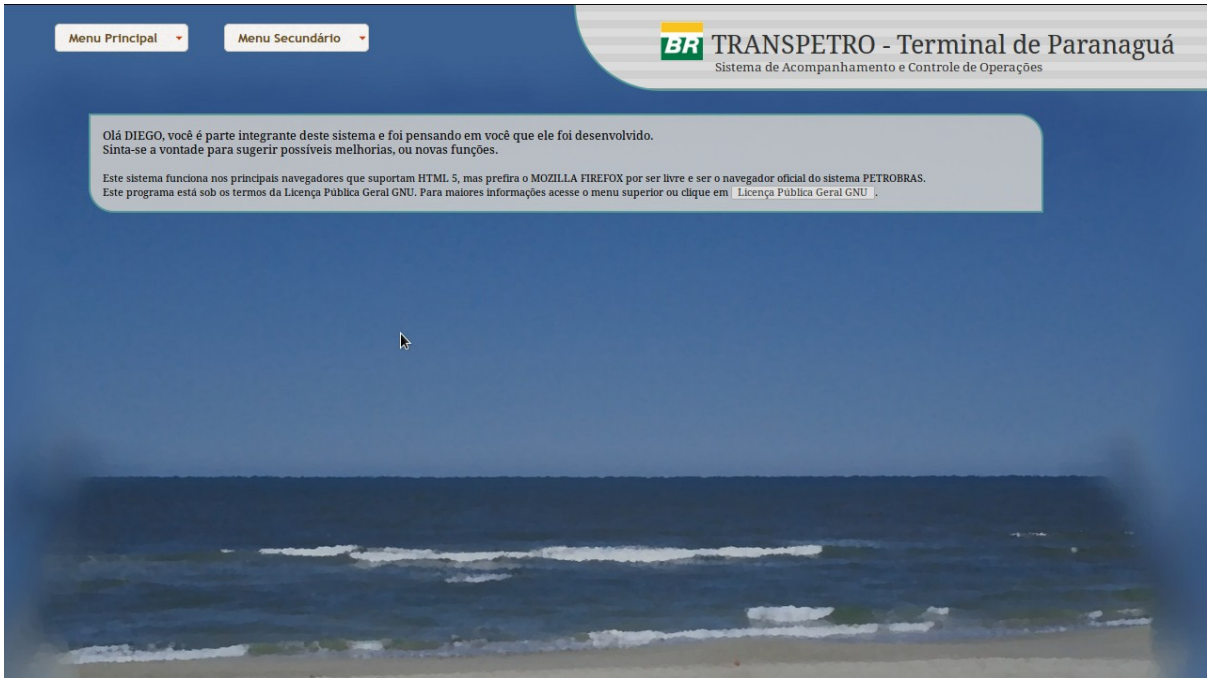
```
  validates_presence_of :data, :vt
  validate :dataVistoriaValidates

  belongs_to :vt

  has_and_belongs_to_many :problemas

  def dataVistoriaValidates
    vistoria = Vt.find(self.vt).vistorias.last
    if self.data != nil and vistoria != nil and vistoria.data > self.data
      self.errors.add :data, "deve ser posterior ou igual a data da última vistoria deste vagão."
    end
    if vistoria != nil and vistoria.id != self.id and  self.id != nil
      self.errors.add :id, ": Você só pode alterar a última vistoria deste vagão."
    end
  end
end
```

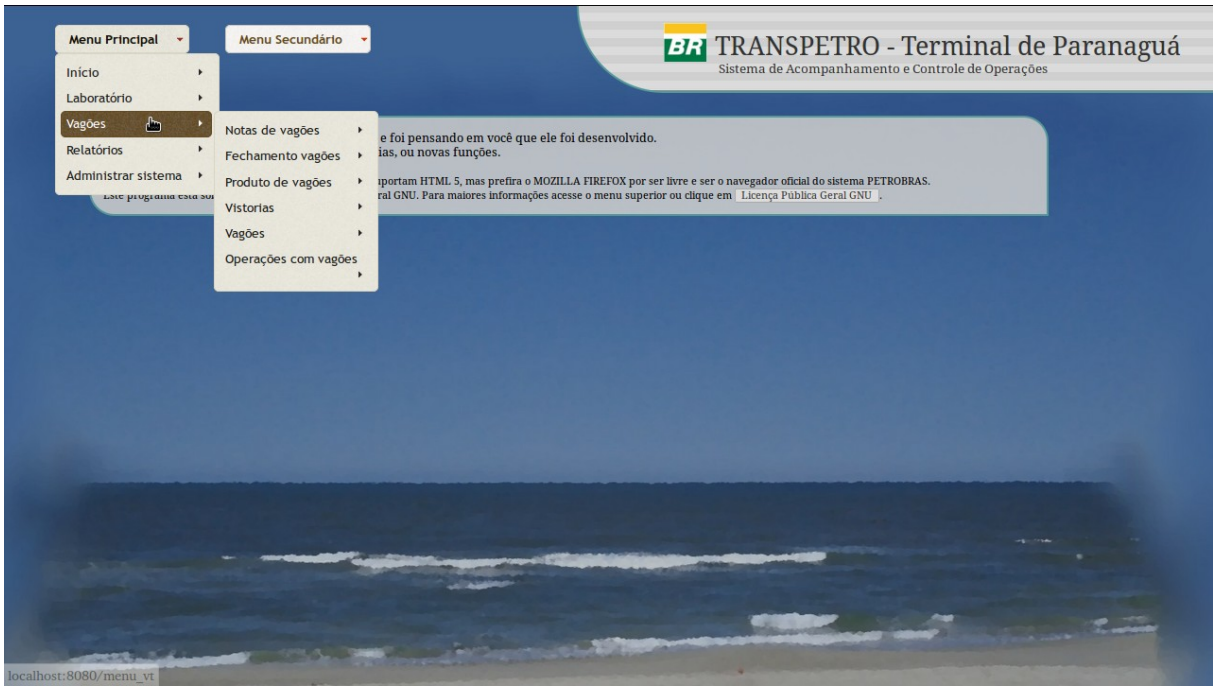
```
#encoding: utf-8
class Vt < ActiveRecord::Base

  validates :numero_curto, :uniqueness => { :case_sensitive => false }, :presence => true,
          :format => {:with => /^([0-9]{4,4})-([0-9]{1,1})$/}
  validates :identificacao_completa, :uniqueness => { :case_sensitive => false }, :presence => true,
          :format => {:with => /^TSC-([0-9]{6,6})-([0-9]{1,1})$/}

  has_many :vistorias
  has_many :operacao_com_vagaos

  def ultimaVistoria
    self.vistorias.last
  end

  private

  def validaFormatoVagao(string_vt)

    case string_vt
    when /^([0-9]{4,4})-([0-9]{1,1})$/; true
    else; false
    end

  end

  def validaFormatoPeso(string_peso)

    case string_peso
    when /^[0-9]{5,5}$/; true
    else; false
    end

  end
  public

  def vtsPorProblema(problema)
    vts = []
```

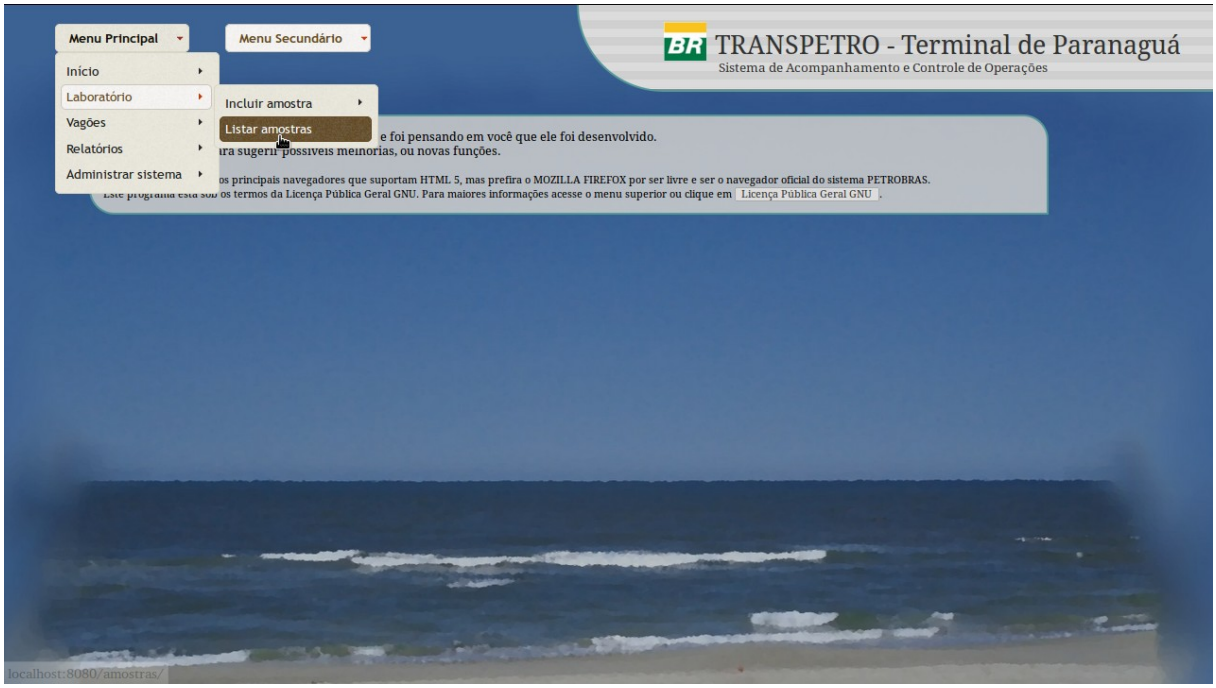
```
        #vagoes = Vt.all
        #Category.all :joins => {:posts => [{:comments => :guest}, :tags]}
        #SELECT categories.* FROM categories INNER JOIN posts ON posts.category_id = categories.id
        #Client.all(:joins => 'LEFT OUTER JOIN addresses ON addresses.client_id = clients.id')
        #EntidadeA.joins(:entidadeB => :entidadeC).where("entidadeC.name = ?", nome)
        #todos vts com ultima vistoria com problema id
        #Student.joins(:schools).where(:schools => { :category => 'public' })
        #vagoes = Vt.joins(:vistorias).where(:vistorias => {:problema => '1'})
        #all :joins => {:vts => {:vistorias => :problemas}}

  vagoes = Vt.all

  vagoes.each do |vt|
    #if vt.vistorias.sort_by{|vistoria| :data}.last.problemas.include?(problema)
    if vt.vistorias.last != nil and vt.vistorias.last.problemas.include?(problema)
      vts << vt
    end
  end
  vts
 end

end
```

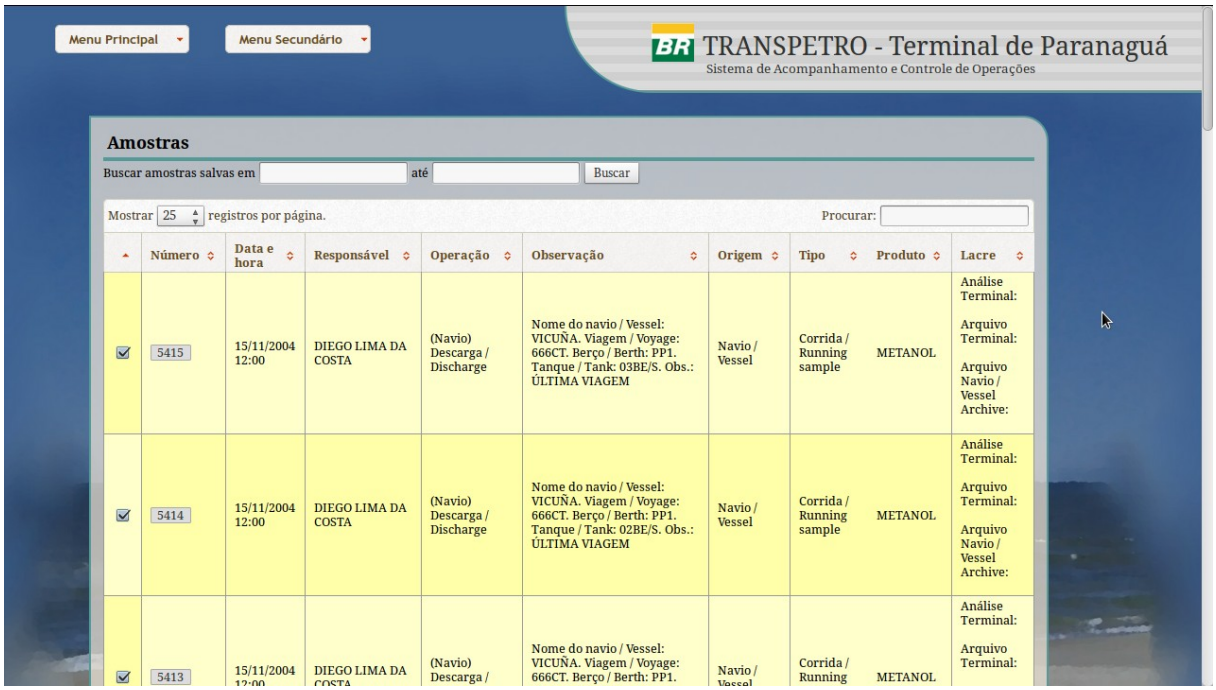
10 APÊNDICE C: IMAGENS DAS TELAS DO SISTEMA

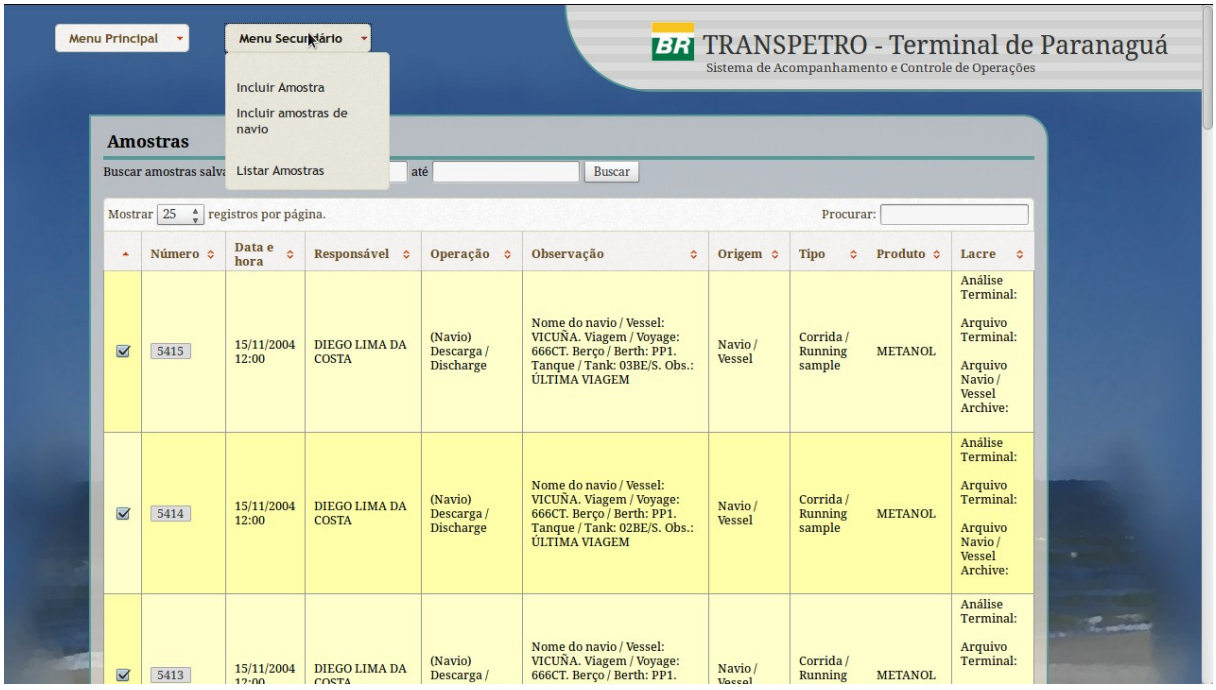


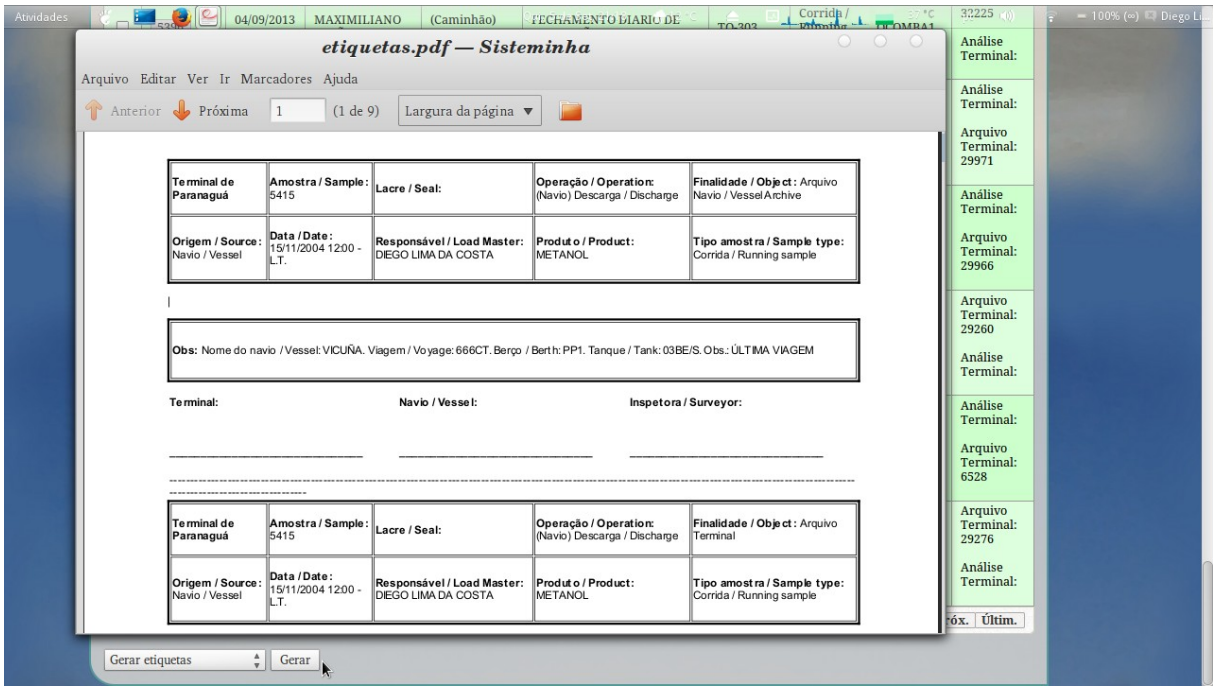


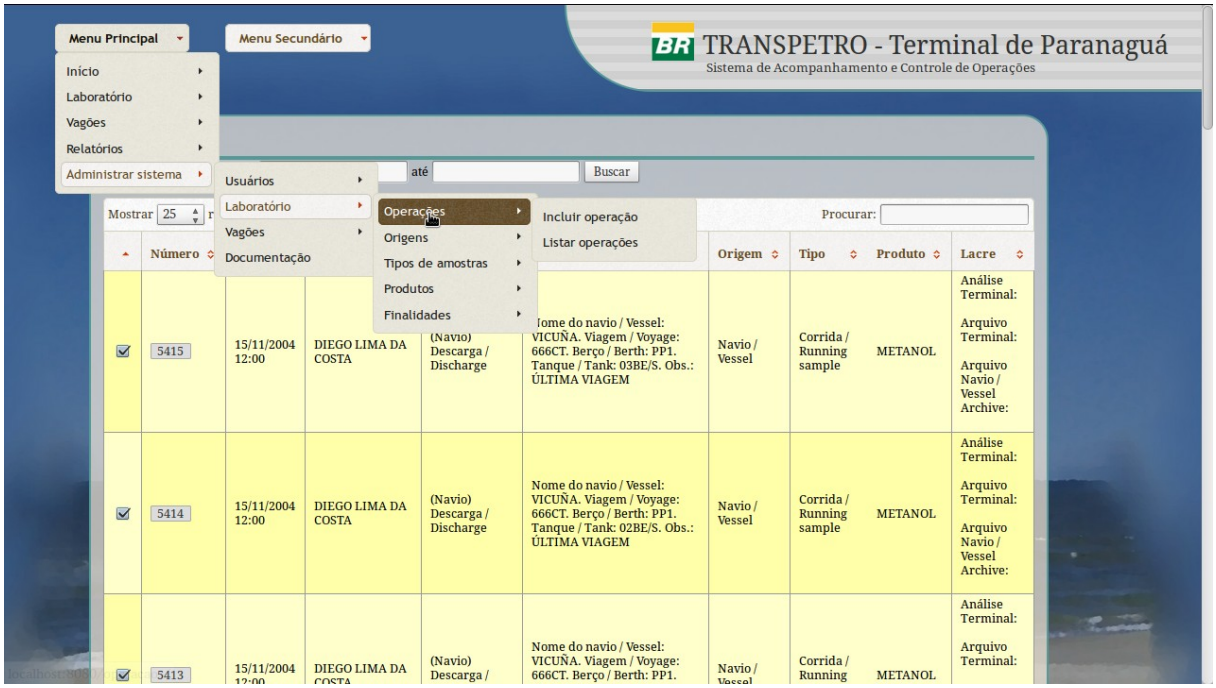

Menu Principal
Menu Secundário
Início
Laboratório
Vagões
Relatórios
Administrar sistema
Notas de vagões
Fechamento vagões
Produto de vagões
Vistorias
Vagões
Operações com vagões
TRANSPETRO - Terminal de Paranaguá
Sistema de Acompanhamento e Controle de Operações
e foi pensando em você que ele foi desenvolvido.
ias, ou novas funções.
portam HTML 5, mas prefira o MOZILLA FIREFOX por ser livre e ser o navegador oficial do sistema PETROBRAS.
ral GNU. Para maiores informações acesse o menu superior ou clique em Licença Pública Geral GNU
localhost:8080/menu_vt

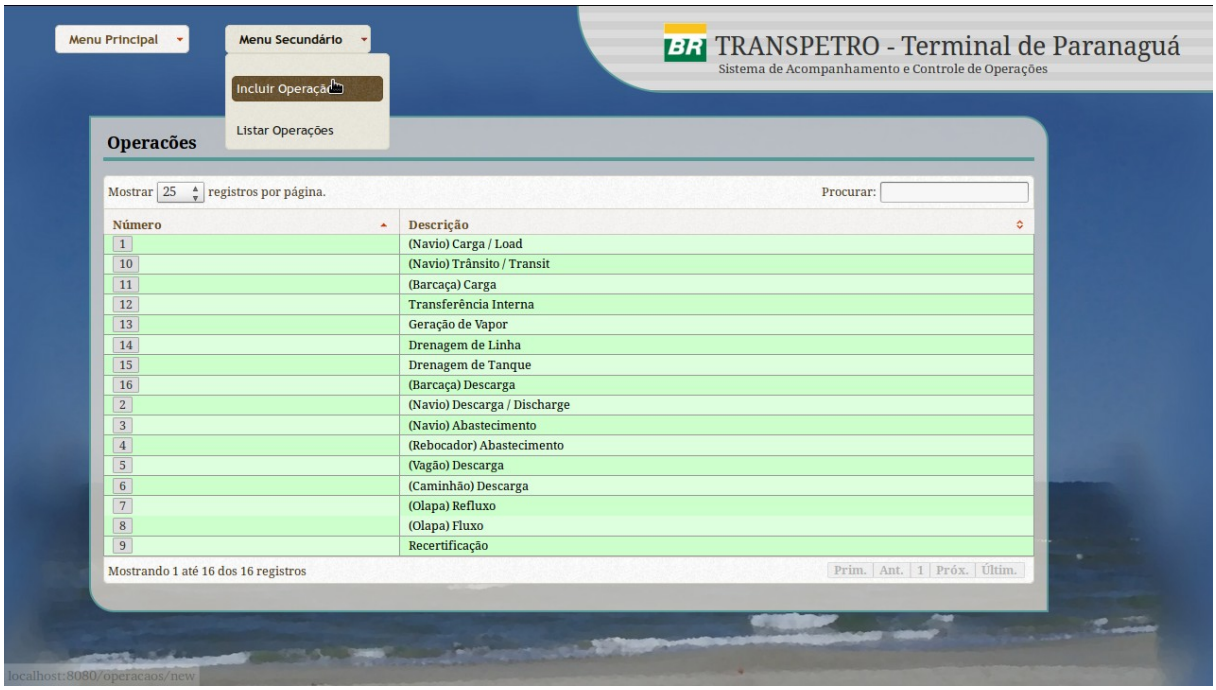

Menu Principal
Menu Secundário
Início
Laboratório
Vagões
Relatórios
Administrar sistema
Incluir amostra
Listar amostras
TRANSPETRO - Terminal de Paranaguá
Sistema de Acompanhamento e Controle de Operações
e foi pensando em você que ele foi desenvolvido.
ias, ou novas funções.
os principais navegadores que suportam HTML 5, mas prefira o MOZILLA FIREFOX por ser livre e ser o navegador oficial do sistema PETROBRAS.
os termos da Licença Pública Geral GNU. Para maiores informações acesse o menu superior ou clique em Licença Pública Geral GNU
localhost:8080/amostras/

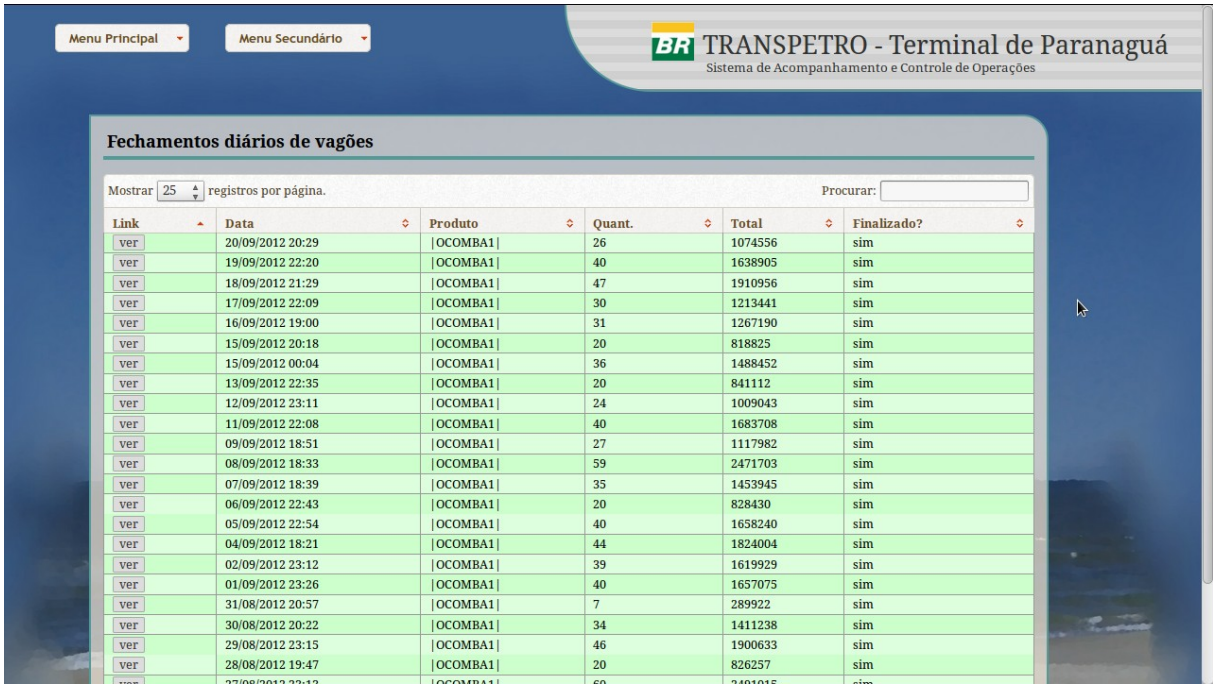

Menu Principal
Menu Secundário
TRANSPETRO - Terminal de Paranaguá
Sistema de Acompanhamento e Controle de Operações
Amostras
Buscar amostras salvas em
até
Buscar
Mostrar 25 registros por página.
Procurar:
Número
Data e hora
Responsável
Operação
Observação
Origem
Tipo
Produto
Lacre
5415
15/11/2004 12:00
DIEGO LIMA DA COSTA
(Navio) Descarga / Discharge
Nome do navio / Vessel: VICUÑA. Viagem / Voyage: 666CT. Berço / Berth: PP1. Tanque / Tank: 03BE/S. Obs.: ÚLTIMA VIAGEM
Navio / Vessel
Corrida / Running sample
METANOL
Análise Terminal:
Arquivo Terminal:
Arquivo Navio / Vessel Archive:
5414
15/11/2004 12:00
DIEGO LIMA DA COSTA
(Navio) Descarga / Discharge
Nome do navio / Vessel: VICUÑA. Viagem / Voyage: 666CT. Berço / Berth: PP1. Tanque / Tank: 02BE/S. Obs.: ÚLTIMA VIAGEM
Navio / Vessel
Corrida / Running sample
METANOL
Análise Terminal:
Arquivo Terminal:
Arquivo Navio / Vessel Archive:
5413
15/11/2004
DIEGO LIMA DA COSTA
(Navio) Descarga /
Nome do navio / Vessel: VICUÑA. Viagem / Voyage: 666CT. Berço / Berth: PP1.
Navio / Vessel
Corrida / Running
METANOL
Análise Terminal:
Arquivo Terminal:

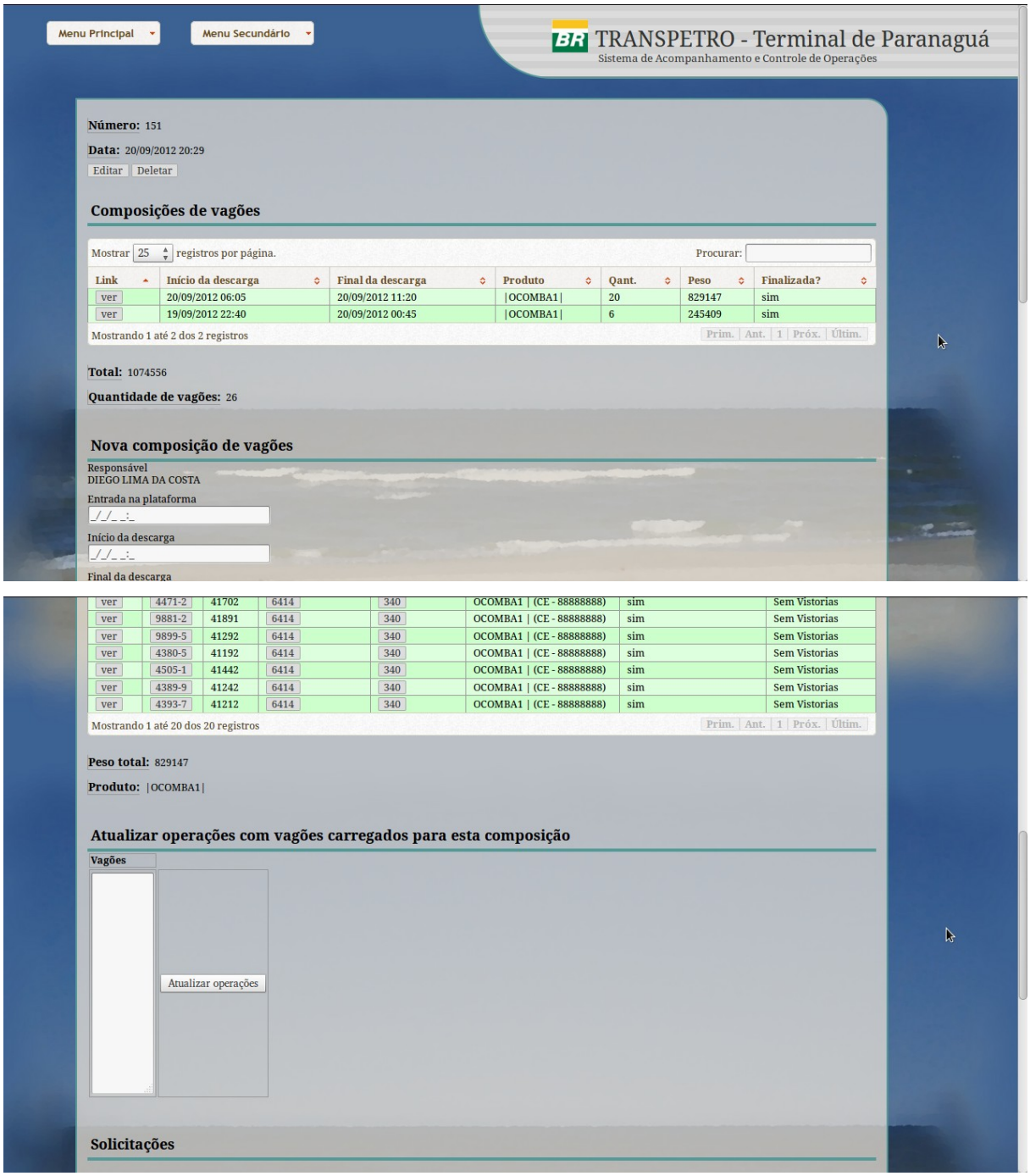

Menu Principal
Menu Secundário
Incluir Amostra
Incluir amostras de navio
Listar Amostras
TRANSPETRO - Terminal de Paranaguá
Sistema de Acompanhamento e Controle de Operações
Amostras
Buscar
Mostrar 25 registros por página.
Procurar:
Número
Data e hora
Responsável
Operação
Observação
Origem
Tipo
Produto
Lacre
5415
15/11/2004 12:00
DIEGO LIMA DA COSTA
(Navio) Descarga / Discharge
Nome do navio / Vessel: VICUÑA. Viagem / Voyage: 666CT. Berço / Berth: PP1. Tanque / Tank: 03BE/S. Obs.: ÚLTIMA VIAGEM
Navio / Vessel
Corrida / Running sample
METANOL
Análise Terminal:
Arquivo Terminal:
Arquivo Navio / Vessel Archive:
5414
15/11/2004 12:00
DIEGO LIMA DA COSTA
(Navio) Descarga / Discharge
Nome do navio / Vessel: VICUÑA. Viagem / Voyage: 666CT. Berço / Berth: PP1. Tanque / Tank: 02BE/S. Obs.: ÚLTIMA VIAGEM
Navio / Vessel
Corrida / Running sample
METANOL
Análise Terminal:
Arquivo Terminal:
Arquivo Navio / Vessel Archive:
5413
15/11/2004
DIEGO LIMA DA COSTA
(Navio) Descarga /
Nome do navio / Vessel: VICUÑA. Viagem / Voyage: 666CT. Berço / Berth: PP1.
Navio / Vessel
Corrida / Running
METANOL
Análise Terminal:
Arquivo Terminal:

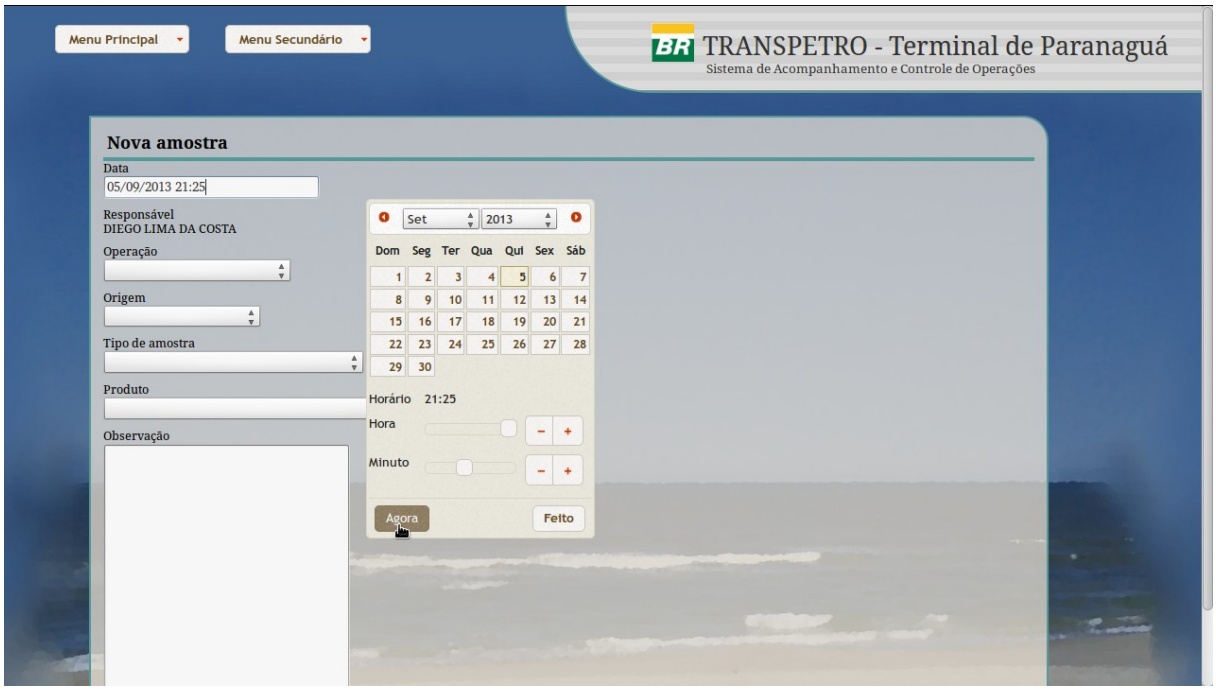

etiquetas.pdf — Sisteminha
Arquivo Editar Ver Ir Marcadores Ajuda
Anterior
Próxima
(1 de 9)
Largura da página
Terminal de Paranaguá
Amostra / Sample: 5415
Lacre / Seal:
Operação / Operation: (Navio) Descarga / Discharge
Finalidade / Object: Arquivo Navio / Vessel Archive
Origem / Source: Navio / Vessel
Data / Date: 15/11/2004 12:00 - L.T.
Responsável / Load Master: DIEGO LIMA DA COSTA
Produto / Product: METANOL
Tipo amostra / Sample type: Corrida / Running sample
Obs: Nome do navio / Vessel: VICUÑA. Viagem / Voyage: 666CT. Berço / Berth: PP1. Tanque / Tank: 03BE/S. Obs.: ÚLTIMA VIAGEM
Terminal:
Navio / Vessel:
Inspetora / Surveyor:
Finalidade / Object: Arquivo Terminal
Gerar etiquetas
Gerar

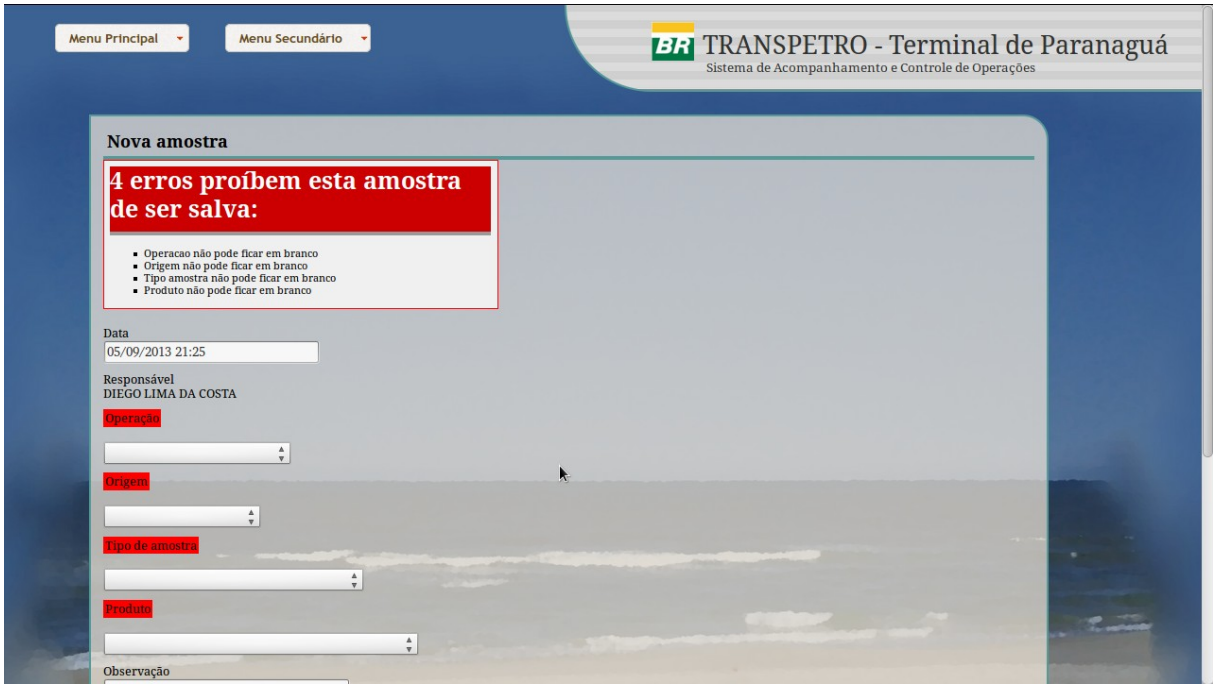

Menu Principal
Menu Secundário
Início
Laboratório
Vagões
Relatórios
Administrar sistema
Usuários
Laboratório
Vagões
Documentação
Operações
Origens
Tipos de amostras
Produtos
Finalidades
Incluir operação
Listar operações
TRANSPETRO - Terminal de Paranaguá
Sistema de Acompanhamento e Controle de Operações
até
Buscar
Mostrar 25
Procurar:
Número
Origem
Tipo
Produto
Lacre
5415
15/11/2004 12:00
DIEGO LIMA DA COSTA
(Navio) Descarga / Discharge
Nome do navio / Vessel: VICUÑA. Viagem / Voyage: 666CT. Berço / Berth: PP1. Tanque / Tank: 03BE/S. Obs.: ÚLTIMA VIAGEM
Navio / Vessel
Corrida / Running sample
METANOL
Análise Terminal:
Arquivo Terminal:
Arquivo Navio / Vessel Archive:
5414
15/11/2004 12:00
DIEGO LIMA DA COSTA
(Navio) Descarga / Discharge
Nome do navio / Vessel: VICUÑA. Viagem / Voyage: 666CT. Berço / Berth: PP1. Tanque / Tank: 02BE/S. Obs.: ÚLTIMA VIAGEM
Navio / Vessel
Corrida / Running sample
METANOL
Análise Terminal:
Arquivo Terminal:
Arquivo Navio / Vessel Archive:
5413
15/11/2004 12:00
DIEGO LIMA DA COSTA
(Navio) Descarga /
Nome do navio / Vessel: VICUÑA. Viagem / Voyage: 666CT. Berço / Berth: PP1.
Navio / Vessel
Corrida / Running
METANOL
Análise Terminal:
Arquivo Terminal:


Menu Principal
Menu Secundário
Incluir Operação
Listar Operações
TRANSPETRO - Terminal de Paranaguá
Sistema de Acompanhamento e Controle de Operações
Operações
Mostrar 25 registros por página.
Procurar:
Número
Descrição
1 (Navio) Carga / Load
10 (Navio) Trânsito / Transit
11 (Barcaça) Carga
12 Transferência Interna
13 Geração de Vapor
14 Drenagem de Linha
15 Drenagem de Tanque
16 (Barcaça) Descarga
2 (Navio) Descarga / Discharge
3 (Navio) Abastecimento
4 (Rebocador) Abastecimento
5 (Vagão) Descarga
6 (Caminhão) Descarga
7 (Olapa) Refluxo
8 (Olapa) Fluxo
9 Recertificação
Mostrando 1 até 16 dos 16 registros
Prim. Ant. 1 Próx. Últim.
localhost:8080/operacaos/new


Menu Principal
Menu Secundário
TRANSPETRO - Terminal de Paranaguá
Sistema de Acompanhamento e Controle de Operações
Fechamentos diários de vagões
Mostrar 25 registros por página.
Procurar:
Link
Data
Produto
Quant.
Total
Finalizado?
ver 20/09/2012 20:29 |OCOMBA1| 26 1074556 sim
ver 19/09/2012 22:20 |OCOMBA1| 40 1638905 sim
ver 18/09/2012 21:29 |OCOMBA1| 47 1910956 sim
ver 17/09/2012 22:09 |OCOMBA1| 30 1213441 sim
ver 16/09/2012 19:00 |OCOMBA1| 31 1267190 sim
ver 15/09/2012 20:18 |OCOMBA1| 20 818825 sim
ver 15/09/2012 00:04 |OCOMBA1| 36 1488452 sim
ver 13/09/2012 22:35 |OCOMBA1| 20 841112 sim
ver 12/09/2012 23:11 |OCOMBA1| 24 1009043 sim
ver 11/09/2012 22:08 |OCOMBA1| 40 1683708 sim
ver 09/09/2012 18:51 |OCOMBA1| 27 1117982 sim
ver 08/09/2012 18:33 |OCOMBA1| 59 2471703 sim
ver 07/09/2012 18:39 |OCOMBA1| 35 1453945 sim
ver 06/09/2012 22:43 |OCOMBA1| 20 828430 sim
ver 05/09/2012 22:54 |OCOMBA1| 40 1658240 sim
ver 04/09/2012 18:21 |OCOMBA1| 44 1824004 sim
ver 02/09/2012 23:12 |OCOMBA1| 39 1619929 sim
ver 01/09/2012 23:26 |OCOMBA1| 40 1657075 sim
ver 31/08/2012 20:57 |OCOMBA1| 7 289922 sim
ver 30/08/2012 20:22 |OCOMBA1| 34 1411238 sim
ver 29/08/2012 23:15 |OCOMBA1| 46 1900633 sim
ver 28/08/2012 19:47 |OCOMBA1| 20 826257 sim


Menu Principal
Menu Secundário
TRANSPETRO - Terminal de Paranaguá
Sistema de Acompanhamento e Controle de Operações
Número: 151
Data: 20/09/2012 20:29
Editar
Deletar
Composições de vagões
Mostrar 25 registros por página.
Procurar:
Link
Início da descarga
Final da descarga
Produto
Qant.
Peso
Finalizada?
ver
20/09/2012 06:05
20/09/2012 11:20
|OCOMBA1|
20
829147
sim
ver
19/09/2012 22:40
20/09/2012 00:45
|OCOMBA1|
6
245409
sim
Mostrando 1 até 2 dos 2 registros
Prim.
Ant.
1
Próx.
Últim.
Total: 1074556
Quantidade de vagões: 26
Nova composição de vagões
Responsável
DIEGO LIMA DA COSTA
Entrada na plataforma
_/_/_ _:_
Início da descarga
_/_/_ _:_
Final da descarga
ver 4471-2 41702 6414 340 OCOMBA1 | (CE - 88888888) sim Sem Vistorias
ver 9881-2 41891 6414 340 OCOMBA1 | (CE - 88888888) sim Sem Vistorias
ver 9899-5 41292 6414 340 OCOMBA1 | (CE - 88888888) sim Sem Vistorias
ver 4380-5 41192 6414 340 OCOMBA1 | (CE - 88888888) sim Sem Vistorias
ver 4505-1 41442 6414 340 OCOMBA1 | (CE - 88888888) sim Sem Vistorias
ver 4389-9 41242 6414 340 OCOMBA1 | (CE - 88888888) sim Sem Vistorias
ver 4393-7 41212 6414 340 OCOMBA1 | (CE - 88888888) sim Sem Vistorias
Mostrando 1 até 20 dos 20 registros
Prim.
Ant.
1
Próx.
Últim.
Peso total: 829147
Produto: |OCOMBA1|
Atualizar operações com vagões carregados para esta composição
Vagões
Atualizar operações
Solicitações


Menu Principal
Menu Secundário
TRANSPETRO - Terminal de Paranaguá
Sistema de Acompanhamento e Controle de Operações
Nova amostra
Data
05/09/2013 21:25
Responsável
DIEGO LIMA DA COSTA
Operação
Origem
Tipo de amostra
Produto
Observação
Set
2013
Dom Seg Ter Qua Qui Sex Sáb
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
Horário 21:25
Hora
Minuto
Agora
Feito


Menu Principal
Menu Secundário
TRANSPETRO - Terminal de Paranaguá
Sistema de Acompanhamento e Controle de Operações
Nova amostra
4 erros proíbem esta amostra de ser salva:
Operacao não pode ficar em branco
Origem não pode ficar em branco
Tipo amostra não pode ficar em branco
Produto não pode ficar em branco
Data
05/09/2013 21:25
Responsável
DIEGO LIMA DA COSTA
Operação
Origem
Tipo de amostra
Produto
Observação