UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

SISTEMA PARA GESTÃO E CONTROLE DE INSPEÇÕES REALIZADAS EM EMBARCAÇÕES

RAMONN MAES ADAMI

BLUMENAU
2012

2012/2-23

RAMONN MAES ADAMI

SISTEMA PARA GESTÃO E CONTROLE DE INSPEÇÕES REALIZADAS EM EMBARCAÇÕES

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Aurélio Faustino Hoppe, Mestre - Orientador

BLUMENAU
2012

2012/2-23

SISTEMA PARA GESTÃO E CONTROLE DE INSPEÇÕES REALIZADAS EM EMBARACAÇÕES

Por

RAMONN MAES ADAMI

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Aurélio Faustino Hoppe, Mestre – Orientador, FURB
Membro:	Prof. Everaldo Artur Grahl, Mestre – FURB
Membro:	Prof. Alexander Roberto Valdameri, Mestre – FURB

Blumenau, 11 dezembro de 2012.

Dedico este trabalho a todos os amigos, familiares em especial a meus pais José e Lúcia e a todos que contribuíram para realização deste.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça, por me dar forças para continuar caminhando em dias nebulosos.

À minha família, que soube suportar todos os meus momentos de fúria devido à pressão imposta no decorrer do curso.

Ao meu irmão, por ter contribuído no desenvolvimento e aplicação da ideia do trabalho.

Aos meus amigos, pelos empurrões e cobranças. A galera do futebol, em especial João, Leandro, Hueliton e Pablo, por fazer com que nestes últimos anos eu não tenha me tornado sedentário.

A minha querida amiga, companheira e namorada Jéssica, por me apoiar em todos os instantes que precisei.

Ao meu orientador, Aurélio Faustino Hoppe, por ter acreditado na conclusão deste trabalho.

A vida é para quem topa qualquer parada. Não para quem pára em qualquer topada.

Bob Marley

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema para automatizar o registro de inspeções realizadas em embarcações. O sistema é responsável por importar e armazenar a estrutura geométrica ou plano de capacidade do barco, permitindo ao usuário acessar os compartimentos da estrutura e selecionar o local para associação da inspeção. A partir desta seleção é possível registrar os dados da inspeção conforme coletado pelo inspetor. O plano de capacidade da embarcação é gerado com ajuda de diretivas da *Application Programming Interface* (API) *Open Graphics Library* (OpenGL) com base nas informações de coordenadas extraídas através da leitura de um arquivo de extensão *Scalable Vector Graphics* (SVG) que segue o padrão *Extensible Markup Language (*XML). Os resultados foram satisfatórios e comprovam que é possível o desenvolvimento de um sistema eficiente quanto a sua capacidade de registrar e associar inspeções em determinados compartimentos de uma embarcação e, consequentemente, obter relatórios e gráficos dos dados armazenados.

Palavras-chave: Embarcação. Inspeção. OpenGl. Controle de qualidade. Arquivo SVG.

ABSTRACT

This work presents the development of system to automate the recording of inspections on vessels. The system is responsible for storing the geometric structure plan or boat capacity, allowing the user to access the compartments of the structure and select the location for inspection association. From this selection you can record the inspection data as collected by the inspector. The vessel capacity plan is generated with the help of Policy Application Programming Interface (API) Open Graphics Library (OpenGL) based on coordinate information extracted by reading a file extension Scalable Vector Graphics (SVG) that follows the pattern Extensible Markup Language (XML). The results were satisfactory and show that it is possible to develop an efficient tool for their ability to record and associate inspections in certain compartments of a vessel and thereby obtain reports and graphs of the data stored.

Keywords: Watercraft. Inspection. OpenGl. Quality of control. SVG file.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API – *Application Programming Interface*

BCC – Curso de Ciência da Computação – Bacharelado

DLL – *Dynamic Link Library*

DML – *Data Manipulation Language*

DDL – *Data Definition Language*

DOM – *Document Object Model*

DSC – Departamento de Sistemas e Computação

GD – Geometria Descritiva

HTML – *HyperText Markup Language*

ISO – *International Organization for Standardization*

IDE – *Integrated Development Environment*

IMO – *International Maritime Organization*

JDom – *Java Document Object Model*

NDC – *Normalized Device Coordinates*

OpenGl – *Open Graphics Library*

PDA – *Personal Digital Assistants*

RF – Requisito Funcional

RNF – Requisito Não-Funcional

SVG – *Scalable Vector Graphics*

SRI – Sistema de Recuperação de Informações

SQL – *Structured Query Language*

UC – *Use Case*

UML – *Unified Modeling Language*

XML - *eXtensible Markup Language*

W3C – *Word Wide Web Consortium*

SUMÁRIO

1 INTRODUÇÃO

Carvalho (1997, p. 72) comenta que a partir da revolução industrial os conhecimentos tecnológicos e a estruturas sociais foram modificados de forma acelerada. Porém, foi a partir da segunda metade do século XX que a humanidade mais acumulou conhecimentos e acelerou o processo de transformações sociais.

Segundo Miranda (2002, p. 11), o principal fator que contribuiu para essas transformações sociais foram as novas tecnologias que causaram e ainda causam um grande impacto na sociedade, impressionando cada vez mais o ser humano. Tais tecnologias podem ser novos processos ou novos softwares que visam melhorar a vida social ou profissional despertando hábitos pessoais e modificando os processos desenvolvidos nas empresas.

A partir destas afirmações nota-se que os softwares estão cada vez mais presentes dentro das empresas devido ao aumento de serviços. Porém, nem sempre atendem aos anseios do consumidor, como por exemplo, os sistemas de inspeção naval.

Os sistemas de inspeção naval contribuem na documentação e distribuição de tarefas exigidas na área da arte naval que segundo Fonseca (1989, p. 25) é o estudo do navio, da sua estrutura, equipamento, conservação e das manobras com que neles se fazem e fainas[1] que nele se realizam. Diversos processos precisam ser registrados e operados de forma a facilitar o entendimento e reduzir o percentual de falhas sobre o mesmo, porém subentende-se uma grande dependência de recursos não apropriados que atrapalham e atrasam os projetos desenvolvidos. Os registros de inspeção naval são partes importantes de estratégias aplicadas por gestores da área de controle de qualidade e que servem para controlar todas as pendências de uma liberação de um barco para navegação (FONSECA, 1989, p.78).

A partir deste cenário, percebeu-se que existe uma grande quantidade de processos relacionados a liberação da embarcação e que influenciam diretamente nos prazos para liberação do barco para navegação. Tais processos vão de encontro ao objetivo do setor de controle de qualidade que é de extrema importância na área naval. O processo mais executado é o registro da inspeção, porém, atualmente existem poucos sistemas que controlam e analisam os dados registrados.

Diante do exposto, desenvolveu-se um sistema para registrar as inspeções de forma interativa a partir da planta ou plano de capacidade da embarcação tornando o trabalho de

[1] Faina é atividade ou trabalho a que concorre ponderável parcela da tripulação de um navio.

registro das inspeções rápido e interativo. É importante ressaltar que o protótipo DeskSI descrito na seção de trabalhos correlatos, foi utilizado como base para criação da tela de registro de inspeções.

1.1 OBJETIVOS DO TRABALHO

Este trabalho tem como objetivo desenvolver um sistema que irá controlar o registro de inspeções em embarcações, facilitando o dia-a-dia dos funcionários e gestores do setor de controle de qualidade.

Os objetivos específicos do trabalho são:

a) ler um arquivo SVG, transformá-lo em coordenadas geométricas afim de reproduzir a estrutura física de uma embarcação;
b) disponibilizar a estrutura geométrica da embarcação em uma interface gráfica utilizando diretivas da API OpenGL;
c) disponibilizar uma interface interativa que possibilite a associação entre a localização do compartimento e a inspeção a ser realizada na embarcação;
d) disponibilizar uma interface para registrar os dados da inspeção que foram coletadas pelo inspetor.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em quatro capítulos, onde no primeiro foi apresentada a introdução, os objetivos e a estrutura do trabalho. O segundo capítulo apresenta a fundamentação teórica, contextualizando os temas abordados no desenvolvimento do trabalho, tais como: arte naval, inspeções, controle de qualidade, métodos de visualização com OpenGL e trabalhos correlatos. No terceiro capítulo é apresentado o desenvolvimento do sistema, contendo desde a especificação até a operacionalidade. O quarto e último capítulo apresentam as conclusões, limitações e algumas propostas de extensões para o sistema.

2 FUNDAMENTAÇÃO TEÓRICA

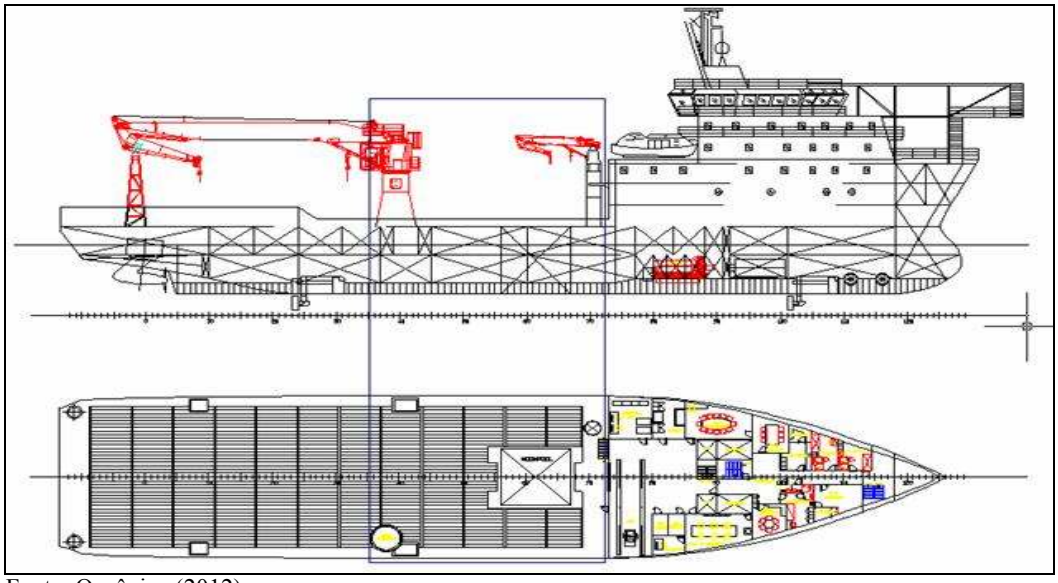
Este capítulo apresenta a fundamentação teórica, contextualizando os conceitos e técnicas necessárias para o desenvolvimento do sistema, tais como: entendimento da arte naval, funções do controle de qualidade, uma visão sobre o que é, e o quão é importante o registro de uma inspeção, métodos de visualização gráfica com OpenGL, geometria computacional, estrutura do arquivo gráfico utilizado, o que é a inspeção e trabalhos correlatos.

2.1 ARTE NAVAL

Arte naval é o ramo responsável por executar atividades que se destinam a exploração das potencialidades do mar. Tem como principal objetivo manipular embarcações e plataformas dos mais variados tipos (SERUNIVERSITÁRIO, 2012).

Embarcação, segundo Ferreira (1986, p. 510) é uma construção feita de madeira, concreto, ferro, aço ou da combinação desses e outros materiais, que flutua e é destinada a transportar pela água pessoas ou cargas. Uma embarcação é dividida em várias partes, dentre essas as mais conhecidas são casco, proa, popa, bordos, meia-nau, bico de proa, a vante e a ré, obras vivas e obras mortas.

Segundo Fonseca (1989, p. 17), assim como em todas as áreas que trabalham com estruturas físicas, a área naval não é diferente e está fundamentada com apoio de desenhos técnicos desenvolvido por ferramentas gráficas. As estruturas geométricas encontradas nesses desenhos consistem em várias formas que desdobram em camadas os compartimentos de uma embarcação, servindo de auxilio técnico no momento de um reparo ou até mesmo da construção da embarcação. Contém os mais variados fatores que tornam os desenhos detalhados e com o maior número de informações possíveis para auxílio do manuseador, nesse caso o funcionário. Com base nessas informações é possível fazer uma leitura identificando os compartimentos de tal embarcação, tornando fácil a identificação de locais específicos conforme mostra a Figura 1.



Fonte: Oceânica (2012).

Figura 1 - Planta da embarcação tipo rebocador

Existem diversas visões sobre a estrutura ou planta e uma delas e a geometria descritiva[2], que mostra a vista ortogonal superior do objeto. O termo também pode se referir às plantas baixas de desenhos técnicos.

Analisando a estrutura através da geometria descritiva têm-se uma sequência de pontos, segmentos de reta e polígonos que formam o desenho idêntico a embarcação física, representando de forma concisa um desenho técnico, utilizado por profissionais desta área.

2.2 CONTROLE DE QUALIDADE

Na indústria naval o setor de controle de qualidade está envolvido diretamente no desenvolvimento dos sistemas que garantem que os serviços sejam executados de forma que supere as expectativas do cliente. Está inteiramente ligado a algumas engenharias como engenharia de qualidade e garantia de qualidade que já existem desde o início das áreas de produção.

Controle de qualidade envolve tipicamente assegurar as conformidades com os padrões mínimos de material e de fabricação, a fim de assegurar o desempenho das instalações de acordo com os desenhos.

---

[2] A Geometria descritiva é um ramo da geometria que tem como objetivo representar objetos de três dimensões em um plano bidimensional. (MANDARINO, 1996).

O controle de qualidade tem como principal objetivo (GENTEPRAIAS, 2011):

a) custos evitáveis: hora de re-trabalho com filosofia baseada em fazer certo na primeira vez, defeitos e falhas nos processos de soldagem e montagem, processamento de reclamações;
b) ênfase na motivação dos funcionários: treinamento de funcionários em determinadas áreas, reconhecimento do trabalho bem feito.

O principal fator que impulsiona a evolução desse setor são as grandes perdas e a maior concorrência no mercado atual onde a qualidade se coloca numa posição como item principal diante dessa concorrência. Para isso a especificação técnica dos funcionários do controle de qualidade deve ser desejável e eles precisam estar em sintonia para compreender os objetivos estratégicos do estaleiro (GENTEPRAIAS, 2011).

Todo esse sistema de qualidade é muito ligado a lucratividade da empresa e das expectativas do cliente e atualmente é associado a uma melhoria contínua (GENTEPRAIAS, 2011).

Na área naval o setor de controle de qualidade além de estar envolvido diretamente no desenvolvimento dos sistemas que garantem a qualidade dos serviços é responsável também pelas inspeções realizadas nas embarcações. Estas inspeções controlam o fluxo de liberação das embarcações para navegação.

2.3 INSPEÇÃO

Segundo Ferreira (1986, p. 776), inspeção é a arte de olhar, observar ou ver. Está relacionado ao cargo de inspetor. Torna-se uma medida necessária a partir de quando se tem elementos que sofrem depreciações devido a fatores externos. Segundo Farias (2010, p. 1) a preservação da integridade de qualquer estrutura depende de antever, mensurar ou evitar riscos. Tendo estas duas afirmações, compreende-se que a inspeção é uma ferramenta utilizada para manter a integridade de objetos. Dependendo da área na qual se aplica a inspeção, ela pode conter fatores diferenciados e podem ser realizadas de várias formas. Uma delas é através de testes de limite e de integridade. Estes testes envolvem profissionais e os mais diversos conglomerados de ferramentas especiais para cada tipo de estrutura e objetos.

O processo de inspeção engloba alguns passos que devem ser seguidos para então chegar ao resultado final, sendo eles: definição do local da inspeção, definição do intervalo de

tempo na qual deve ser inspecionado aquele objeto e a definição de como inspecionar. Esses passos são encontrados em qualquer área na qual seja necessária a inspeção e, segundo Farias (2010, p. 17) a melhor forma de resolver esses passos seria manter o pessoal com mais experiência adquirida ou através da sistematização da aplicação do conhecimento acumulado e pelas experiências aliada as técnicas modernas de análise e avaliação estrutural com modelos matemáticos e formulações probabilísticas. Esses processos estão totalmente dependentes da estrutura física e são efetuados com base no desenho técnico ou planta da mesma.

A inspeção em si é uma atividade de revisão de detalhes e que pode gerar um novo processo que é a manutenção. Ferreira (1986, p. 889) define manutenção como o ato ou efeito de manter conservado algum objeto. É um processo que é dependente e gerado através das inspeções e pode ser classificada como preventiva ou corretiva.

2.4 VISUALIZAÇÃO DA INFORMAÇÃO

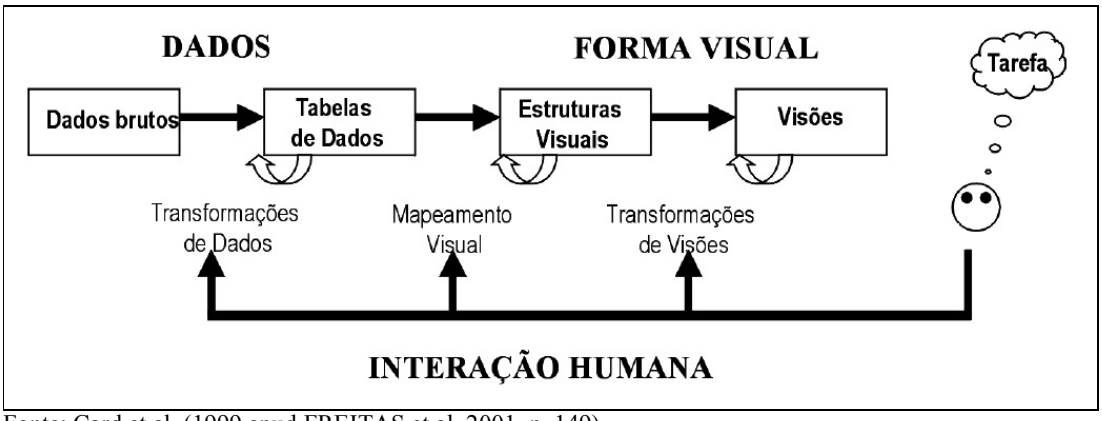
Um conjunto de informações disponíveis na forma textual, gráfica ou tabular, tem como objetivo facilitar o seu entendimento. A forma visual ajuda na interpretação e agiliza a análise dos dados.

> A visualização de dados consiste em disponibilizar informações em uma forma textual, gráfica ou tabular, sendo que o seu principal objetivo é possibilitar a interpretação visual de informações por pessoas. Bons métodos de visualização necessitam que os dados sejam convertidos em um determinado formato, sendo que suas características e relacionamentos com itens ou atributos devem possibilitar boas análises e relatórios. (TAN; STEINBACH; KUMAR, 2006, p. 6).

Segundo Vieira e Corrêa (2010), as informações representadas através de recursos visuais como recursos gráficos e interfaces amigáveis, proporcionam uma melhor compreensão do usuário na recuperação das informações mais relevantes. Existem Sistemas de Recuperação de Informação (SRI) que auxiliam na recuperação de grandes volumes de dados, apresentados de diversas fontes e formatos nos sistemas, com o objetivo de proporcionar uma melhora na compreensão dos usuários na recuperação das informações relevantes, os dados podem ser apresentados através de recursos gráficos e de interfaces amigáveis.

Para o desenvolvimento de sistemas de visualização, os projetistas sempre analisam como expor as informações graficamente para que facilite a interpretação do usuário,

fornecendo-os meios que possam limitar a quantidade de informações, manipulá-las de forma geométrica, ou seja, possibilitando o *zoom* na representação gráfica e também analítica, possibilitando a redução ou expansão do conjunto de dados exibidos de acordo com critérios estabelecidos pelo usuário (FREITAS, 2001, p. 145). A Figura 2 mostra um exemplo de visualização e transformação de dados brutos em uma forma visual com o objetivo de facilitar o entendimento da tarefa do usuário.



Fonte: Card et al. (1999 apud FREITAS et al. 2001, p. 149).

Figura 2 - Modelo de referência para visualização

Para Freitas et al.(2001 apud VIEIRA; CORRÊA, 2010), a visualização da informação é uma "área de aplicação de técnicas de computação gráfica, geralmente interativas, visando auxiliar o processo de análise e compreensão de um conjunto de dados, através de representações gráficas manipuláveis."

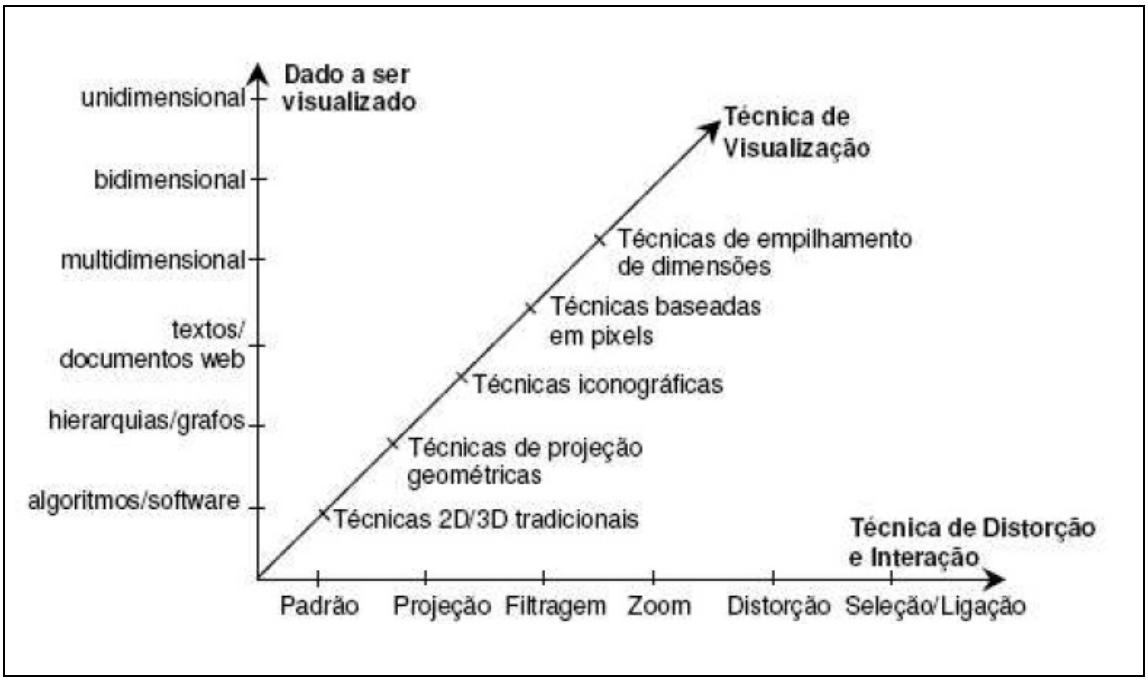
2.4.1 Técnicas de visualização

Escolher uma técnica para ser usada na visualização de um conjunto de dados, não é uma tarefa fácil. Para Freitas (2001, p. 145), considerar o tipo de informação que esta sendo tratada e a tarefa que o usuário deseja realizar, pode ser o inicio para a escolha de uma técnica.

> Shneiderman classificou as técnicas de visualização por tipo de dados e por tarefas: técnicas podem ser unidimensionais (1D), temporais, bidimensionais (2D), tridimensionais (3D) e multidimensionais (nD), dirigidas à visualização de hierarquias e de relacionamentos (grafos), e podem suportar tarefas como a obtenção de uma visão geral, obtenção de visão detalhada, zooming, filtragem, identificação de relacionamentos, manutenção de histórico de ações e extração de informações diversas. (SHNEIDERMAN, 1996, p. 336 apud FREITAS, 2001, 145).

Keim (2002, apud SILVA, 2008, p. 81) sugere três critérios para classificar as técnicas

de visualização, ilustrados na Figura 3: “a natureza do dado a ser visualizada, a abordagem de mapeamento adotada pela técnica e os métodos de interação e distorção usados para manipular a representação visual.”



Fonte: Keim (2002 apud SILVA, 2008, p. 80).

Figura 3 - Classificação das técnicas de visualização

Como foi visto existem várias técnicas de visualização. A seguir será descrita uma técnica de visualização que é utilizada para visualizar dados utilizando a Geometria Descritiva (GD).

2.4.2 Geometria descritiva e habilidade de visualização espacial

Segundo Olkun (2003), a habilidade de visualização espacial pode ser melhorada através de atividades apropriadas. O desenho de engenharia, que é fundamentado pela GD, é utilizado visando essa finalidade por duas razões: primeiro, constitui-se numa base prática de situações reais por meio da representação e visualização de objetos e, segundo, experiências concretas com objetos geométricos e suas representações auxiliam no desenvolvimento da visualização espacial das pessoas.

De acordo com Barros e Santos (2000), o processo de resolução de um problema de GD pode ser dividido em três fases: visualização (processo mental 2D à 3D), concepção (conceitual 3D) e operacionalização (procedimental 2D). A fase de visualização requer

basicamente a compreensão da técnica de representação projetiva e a habilidade de visualização espacial. A fase de concepção caracteriza-se por ser a mais complexa, pois exige raciocínio espacial e abstrato, além da criatividade na busca de soluções ao problema. Nesta etapa é criada e definida a estratégia de solução do exercício. Por fim, a fase de operacionalização envolve o conhecimento dos procedimentos de manipulação dos elementos bidimensionais da épura mongeana e dos teoremas e conceitos básicos que fundamentam esta representação gráfica. Através desses procedimentos e conceitos, a solução concebida na fase anterior é implementada no plano bidimensional.

Aplicando a geometria computacional nos dados obtidos após cumprir as três fases que resolvem um problema de GD é possível obter resultados importantes para análise de processos dos mais variados tipos, inclusive nos processos de inspeção naval.

2.5 GEOMETRIA COMPUTACIONAL

A geometria computacional é o sub-campo da teoria dos algoritmos e envolve o desenvolvimento e análise de algoritmos eficientes para problemas envolvendo o processamento de entradas e saídas geométricas. Um dos objetivos da geometria computacional é prover ferramentas de geometria básicas, para que cada área do conhecimento possa construir seus programas (MOUNT, 2002, p. 2-3). De acordo com Chen (1996, p. 2), existe um grande número de áreas de aplicação, como padrão de reconhecimento, gráficos de computador, processamento de imagem, pesquisa de operações, estatísticas, entre outros, têm sido a incubadora desta disciplina desde que elas proveram problemas geométricos inerentes.

Segundo Fernandes (2009, p. 1) a geometria computacional permite obter e projetar novos elementos geométricos a partir de construções elementares. Alguns problemas que podem ser solucionados a partir da geométrica computacional são: fechos convexos; problemas de proximidade; partições convexas; busca geométrica; e problemas de intersecção.

2.6 ARQUIVOS GRÁFICOS

Segundo Levine (1993, p. 18) um arquivo gráfico é nada mais que um arquivo que armazena uma figura.

Os arquivos gráficos para armazenar figuras de duas dimensões (2D), podem ser do tipo *raster* ou vetorial. Sendo que o primeiro armazena cada pixel da imagem em formato seqüencial e o segundo armazena as equações matemáticas que descrevem a geometria da imagem.

Segundo Morrison (1995), a desvantagem dos arquivos vetoriais é que eles não representam muito bem as imagens de tons contínuos. As imagens de tons contínuos contêm tons em escala entre preto até branco ou de uma cor para outra. Portanto, eles são adequados para desenhos de caracteres e ilustrações técnicas. Os formatos *raster*, por sua vez, funcionam bem para as imagens de tons contínuos, mas se aumentadas demais, a figura passa a ser visualizada como se fosse formada por um conjunto de pequenos quadrados (MORRISON, 1995).

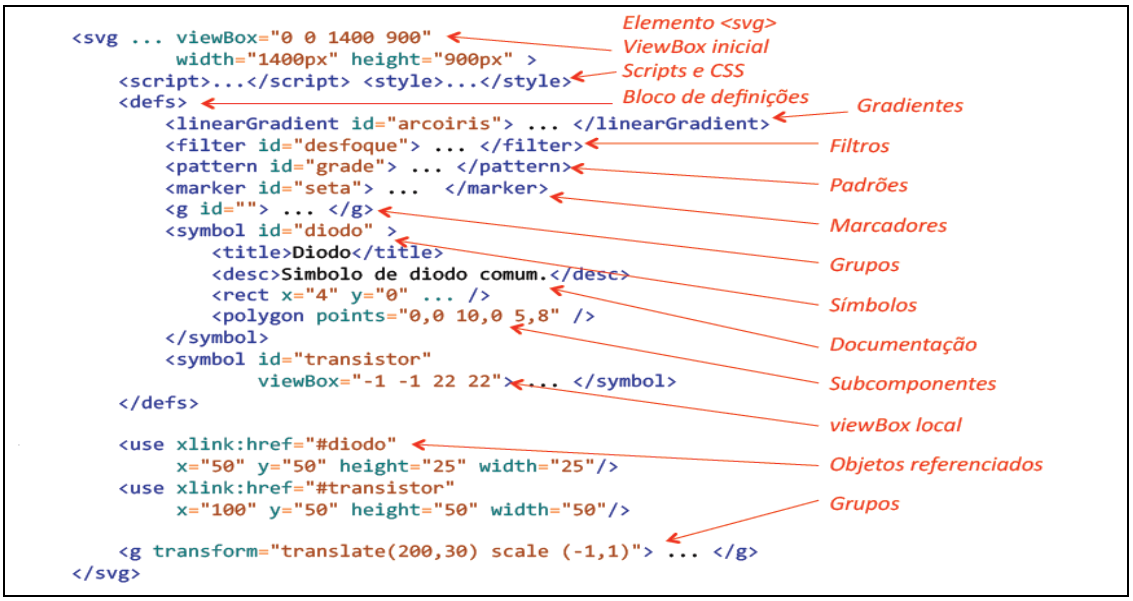
2.6.1 Arquivo SVG

SVG é a abreviatura de *Scalable Vector Graphics* que pode ser traduzido do inglês como gráficos vetoriais escaláveis. Trata-se de uma linguagem XML para descrever de forma vetorial desenhos e gráficos bidimensionais, quer de forma estática, quer dinâmica ou animada. Foi criado e submetido ao *World Wide Web Consortium* (W3C) em 1998, com intuito de melhorar e facilitar a manipulação de imagens vetorizadas na WEB (W3C, 2012). Tudo está expresso em XML, tornando-se fácil de manusear, com a possibilidade de embutir o código em *HyperText Markup Language* (HTML).

O surgimento dos arquivos SVG iniciou novas possibilidades para um desenvolvedor JAVA, possibilitando a criação de interfaces poderosas com animações e outros efeitos gráficos (RAMOS, 2005, p. 9). Com esse formato de arquivo é possível desenvolver gráficos, linhas, polígonos, figuras, textos, filtros e efeitos. É escalável com funções de *zoom* eficiente e rápido sem perda de qualidade. Armazena equações de gráficos e não mapas de pixel, tornando a qualidade da imagem melhor. São imagens com tamanho reduzido se comparando

a outros formatos além de fornecer características importantes como manutenção da qualidade, mesmo quando a imagem é redimensionada.

A Figura 4 mostra a estrutura do arquivo SVG, ao qual é representada por marcadores XML, sendo que os elementos constantes na imagem definem os objetos desenhados pelo software que está lendo o arquivo, sendo especificados como figuras vetoriais (retângulos, círculos, elipse, linha reta, linha com múltiplos segmentos e polígonos).



Fonte: Argonavis (2011).

Figura 4 - Elementos estruturais do arquivo SVG

2.7 OPENGL

Segundo Shreiner et al (2005, p. 13), a *Open Graphics Library* (OpenGL) é uma biblioteca que faz a interface para o hardware gráfico. Esta interface consiste em aproximadamente 150 comandos distintos que podem ser usados para especificar objetos e operações para produzir aplicações interativas em duas ou três dimensões.

> O desenvolvimento de aplicações gráficas de alto desempenho costumava ser exclusividade de instituições de pesquisa ou empresas que dispunham de hardware gráfico veloz e especificado, além de programadores experientes. Há alguns anos, essa situação modificou-se com o surgimento do mercado de placas gráficas para computadores pessoais e o desenvolvimento de bibliotecas gráficas que não exigiam conhecimentos extensivos de programação ou de hardware. Uma dessas bibliotecas, hoje o padrão da indústria para aplicações profissionais, é a OpenGL. (COHEN; MANSSOUR, 2006, p. 15).

Pode-se definir a OpenGL como uma especificação aberta e multiplataforma de uma

biblioteca de rotinas gráficas e de modelagem utilizada para o desenvolvimento de aplicações de Computação Gráfica, tais como jogos ou sistemas de visualização. A maior vantagem da OpenGL é a velocidade, uma vez que incorpora vários algoritmos otimizados, incluindo o desenho de primitivas gráficas, o mapeamento de textura e outros efeitos especiais (COHEN; MANSSOUR, 2006, p. 15). Segundo McReynolds e Blythe (2005, p. 24), a OpenGL pode ser obtida gratuitamente para quase todas as arquiteturas hoje existentes: Apple Machintosh, Microsoft Windows e praticamente todos Unix com algumas variantes incluindo Linux e OS/2.

2.8 TRABALHOS CORRELATOS

A seguir serão apresentadas duas ferramentas relacionadas a automatização dos registros de inspeções: DeskSI – protótipo utilizado para registrar inspeções e o sistema ElcoShip (ELCOSHIP, 2008) criado para atender as normas da *International Maritime Organization* (IMO).

2.8.1 DeskSI - Protótipo

Como já mencionado anteriormente, esse trabalho será uma continuação de um protótipo em uso, ao qual será descrito desta seção.

O desenvolvimento do sistema contou com a contribuição de um funcionário de determinado estaleiro onde após o levantamento das principais dificuldades do gestor para o bom desempenho das atividades no setor, notou-se que eram utilizadas algumas planilhas excel para registro das informações dos funcionários no momento da inspeção e isso tornava o processo de inspeção lento e totalmente dependente do apoio do gestor. O protótipo basicamente conta com uma tela para cadastro de informações das inspeções, que são armazenadas em arquivos serializados. A navegação é feita através de menus e a interface encarrega-se de bloquear informações que não são necessárias, dependendo do tipo da inspeção. O usuário tem a função de selecionar a embarcação e o tipo de inspeção desejado através do componente de lista e então informar os dados recolhidos na inspeção efetuada. Os

principais botões da tela têm funções de criação, gravação e exclusão das informações das inspeções. Foi criado utilizando arquitetura padrão MVC[3] e linguagem orientada a objetos JAVA com apoio da *Integrated Development Environment* (IDE) NetBeans. Não utiliza banco de dados para armazenar os dados e tem por objetivo liberar o gestor das atividades de digitação e compreensão dos dados informados pelos funcionários e também facilitar a demonstração de dados aos seus superiores. A Figura 5 mostra a tela principal do sistema DeskSI.

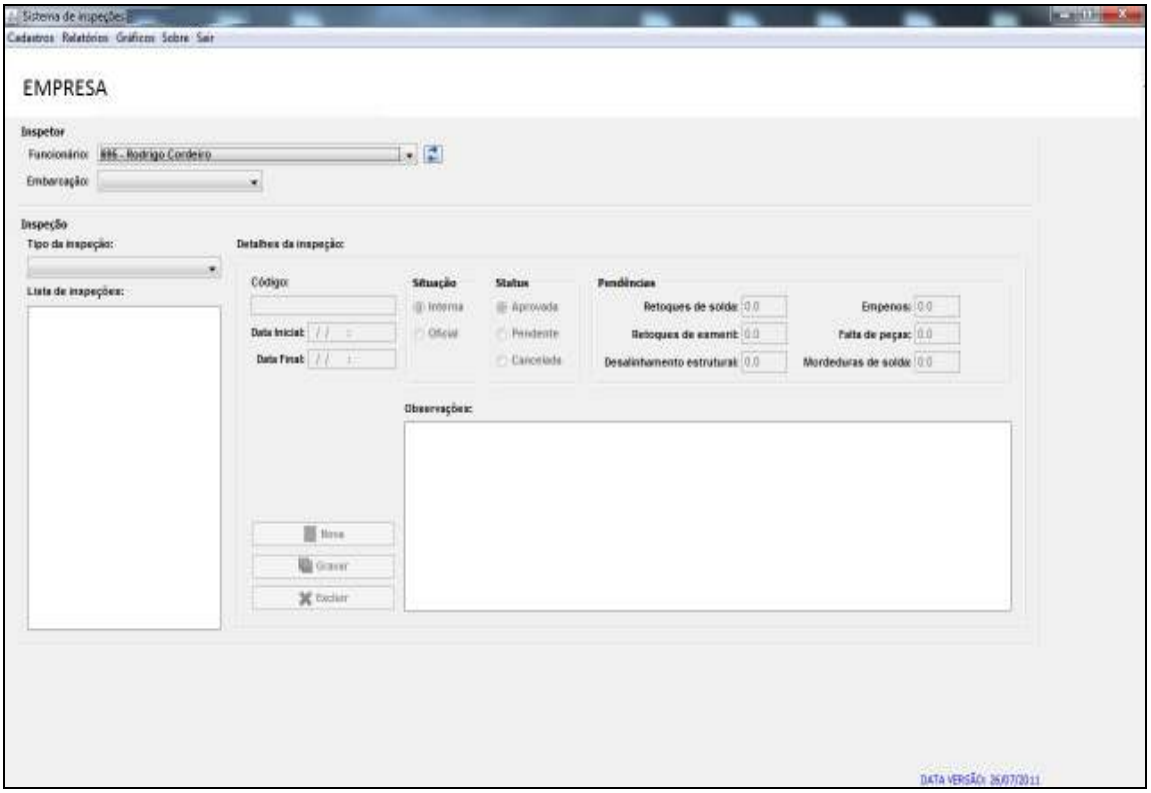


Figura 5 - Tela DeskSI - protótipo

Além da interface exibida na Figura 5, existe o cadastro de funcionário e embarcação, pois são pré-requisitos para o registro da inspeção. As interfaces para geração de relatórios e gráficos são acessadas através dos menus e possuem filtros para facilitar a visualização.

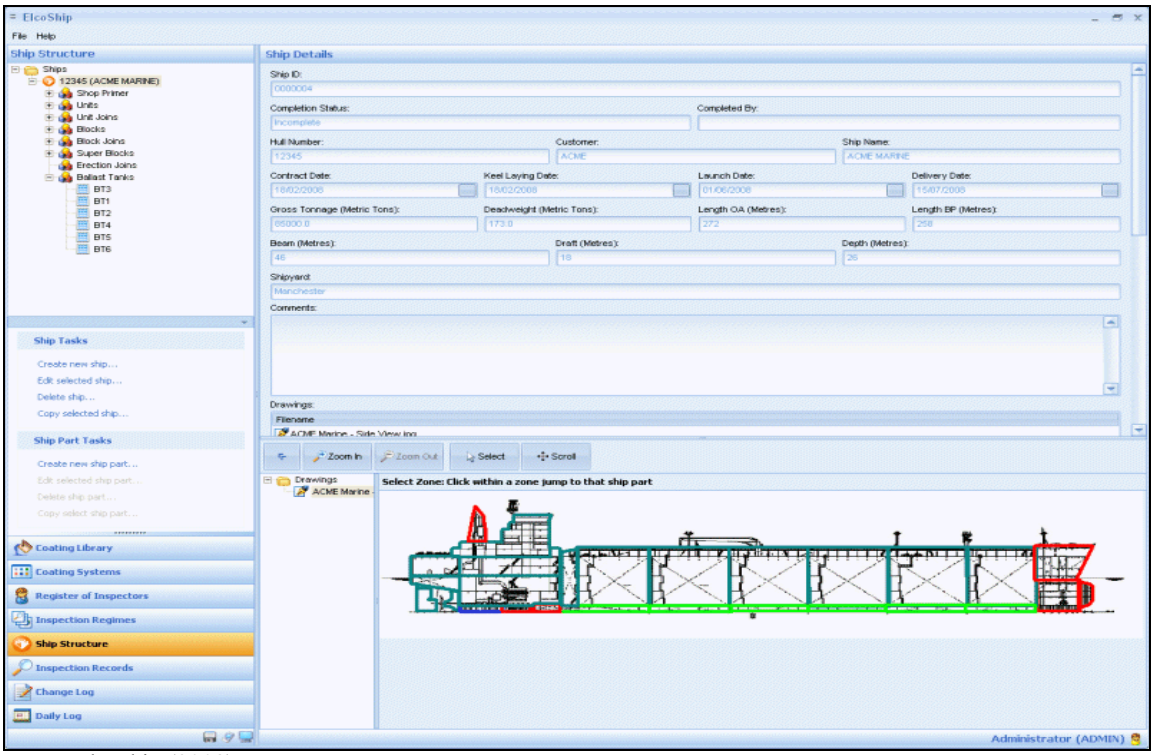
[3] *Model-View-Controller* (MVC) é um modelo de desenvolvimento de software, atualmente considerado uma "arquitetura padrão" utilizada na engenharia de software.

2.8.2 ElcoShip®

ElcoShip (ELCOSHIP, 2008) é um software criado para atender os regulamentos *da* IMO, onde os estaleiros registram para cada área considerada tanto a nível de bloco como os estágios das inspeções em revestimentos de tanques.

O software está dividido em módulos, dos quais os mais importantes são:

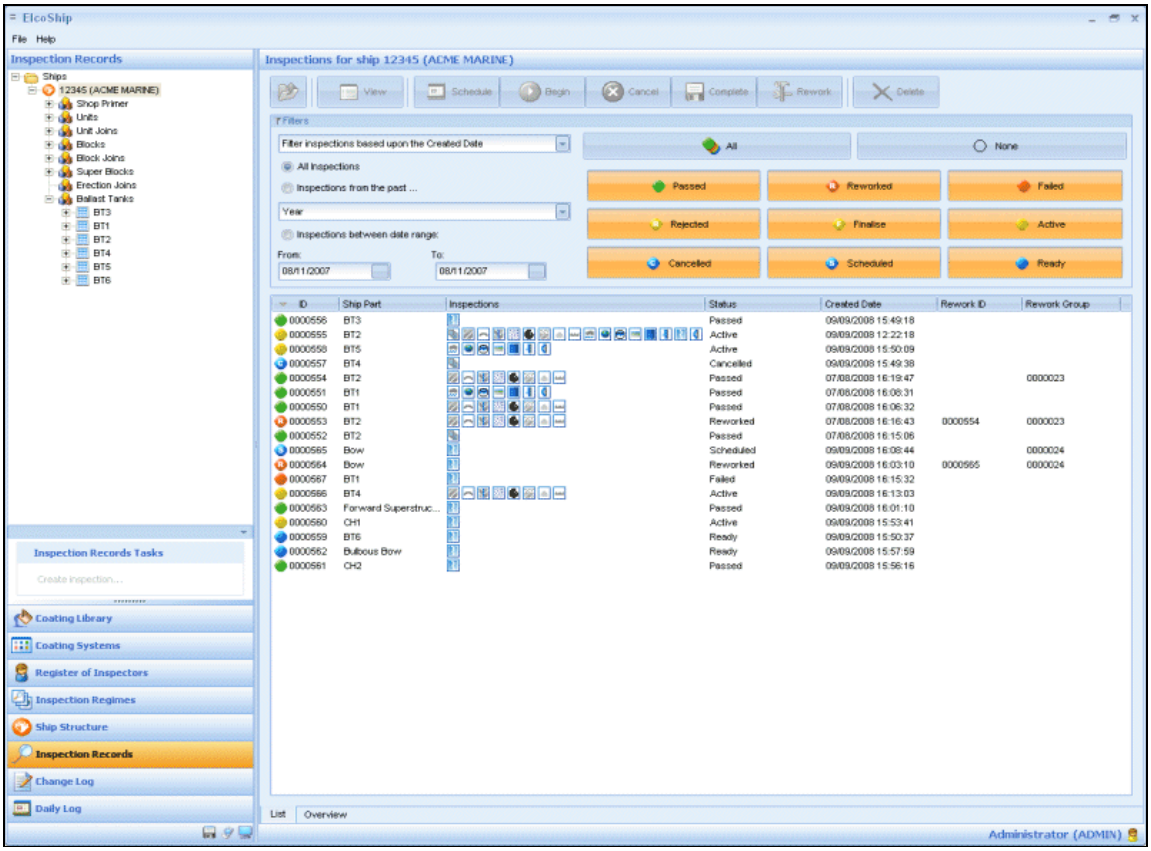
a) estrutura do navio (Figura 6): mostra os desenhos utilizando diagramas de blocos e pode ser configurado para os usuários visualizarem os estados e status de cada inspeção;



Fonte: Elcoship (2008).

Figura 6 - Estrutura do navio no sistema ElcoShip

b) registro de inspeções (Figura 7): registra detalhes de todos os inspetores em conjunto com suas certificações, assinatura, direitos de acesso, documentação pessoal entre outras informações. Os registros são criados por tarefa, a data de área, em análise e por inspetor juntamente com qualquer exigência para o retrabalho ou concessões.



Fonte: Elcoship (2008).

Figura 7 - Registro de inspeções no sistema ElcoShip

Embora o software atenda todas as medidas exigidas pelo IMO, algumas tarefas são visuais e podem ser registradas com o uso da versão *Personal Digital Assistants* (PDA) sobre a plataforma Windows Mobile exibido na Figura 8, onde os usuários podem programar e enviar tarefas de inspeção, gerar dados de inspeção visual ou manual e realizar tarefas programadas a partir do campo.

Fonte: Elcoship (2008).

Figura 8 - ElcoShip versão PDA

Contudo, o ElcoShip procura minimizar a quantidade de relatórios escritos, a maximização do tempo de inspeção no local e a redução de custos para a instituição. Para isso o sistema rastreia em tempo real todos os status de todas as inspeções de revestimentos em todos os navios que estão cadastrados no sistema. (ELCOSHIP, 2008).

3 DESENVOLVIMENTO

A apresentação do desenvolvimento do trabalho está divida em quatro seções. Na primeira seção (3.1) são abordados os principais requisitos definidos para o desenvolvimento do trabalho. Na segunda seção (3.2) é apresentada a especificação do software desenvolvido. Na terceira seção (3.3) é mostrada a implementação do sistema. Na quarta seção (3.4) são abordados os resultados e discussões do trabalho.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Nesta seção são apresentados os requisitos utilizados no desenvolvimento do sistema:
O sistema deve:

a) permitir gerar a estrutura geométrica da embarcação a partir de um arquivo SVG (Requisito Funcional (RF) 01);
b) permitir selecionar o compartimento que sofrerá a inspeção mediante a interface interativa (RF 02);
c) permitir o cadastro de projetos (RF 03);
d) permitir o cadastro de estruturas (RF 04);
e) permitir o cadastro de compartimentos (RF 05);
f) permitir o cadastro de inspeções (RF 06);
g) permitir importar o arquivo contendo as informações da estrutura (RF 07);
h) permitir o cadastro de usuários (RF 08);
i) permitir o cadastro de tipo de inspeção (RF 09);
j) permitir o cadastro de pendências (RF 10);
k) permitir gerar gráficos utilizando filtros de data, funcionário, estrutura, projeto e compartimento (RF 11);
l) permitir gerar relatórios utilizando filtros de data, funcionário, estrutura, projeto e compartimento (RF 12);
m) permitir selecionar o compartimento (RF 13);
n) funcionar no sistema operacional Windows win32 ou superior (Requisito não funcional (RNF) 01);

o) ser implementado em linguagem Java (RNF 02);
p) utilizar banco de dados MySQL (RNF 03);
q) utilizar biblioteca gráfica OpenGL para renderizar a estrutura importada (RNF 04).

3.2 ESPECIFICAÇÃO

O sistema foi especificado utilizando diagramas da *Unified Modeling Language* (UML) usando orientação a objetos. Foi utilizada a ferramenta *Enterprise Architect* (SPARXSYSTEMS, 2000) para o desenvolvimento dos diagramas de caso de uso, de classe e de sequência. Na seção 3.2.1 é apresentado o diagrama de casos de uso, na seção 3.2.2 são apresentados os diagramas de classes, na seção 3.2.3 são apresentados os diagramas de sequência e na seção 3.2.4 é apresentado o modelo entidade relacionamento,

3.2.1 Diagrama de caso de uso

A Figura 9 exibe o diagrama de caso de uso utilizado para desenvolvimento do trabalho. São identificados onze casos e dois atores envolvidos (funcionário e gestor).

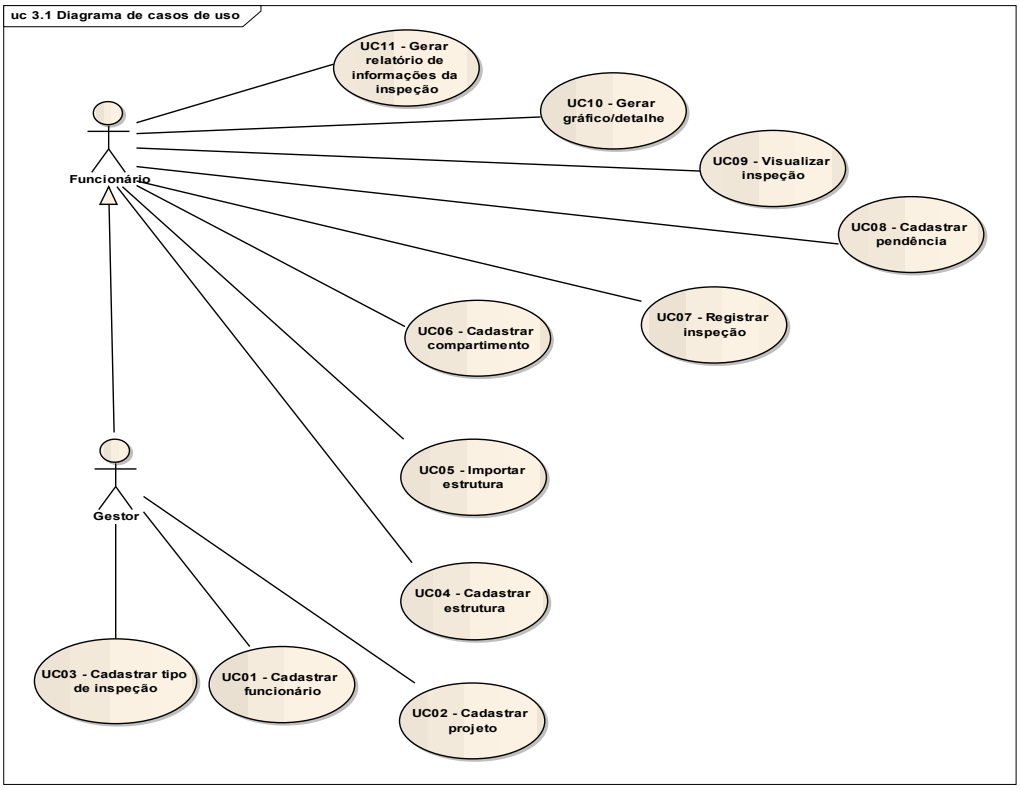


Figura 9 - Diagrama de casos de uso

No caso de uso `UC 01 – Cadastrar funcionário`, o usuário gestor efetuará o cadastro de um novo funcionário, liberando o acesso ao sistema;

No caso de uso `UC 02 – Cadastrar projeto`, o usuário gestor fará o cadastro de um novo projeto ou então realizará alterações no projeto desejado.

No caso de uso `UC 03 – Cadastrar tipo de inspeção`, o usuário gestor fará o cadastro de um novo tipo de inspeção ou então realizará a alteração em um já existente. Esta funcionalidade é pouco utilizada, pois os tipos de inspeção dificilmente mudam.

No caso de uso `UC 04 – Cadastrar estrutura`, o usuário fará o cadastro de uma nova estrutura ou então realizará alterações na estrutura desejada. Este cadastro está disponibilizado para qualquer usuário e poderá ser utilizado somente após a seleção de um projeto.

No caso de uso `UC 05 – Importar estrutura`, o usuário utilizará o botão existente no frame principal para selecionar o arquivo desejado, utiliza a API padrão do Windows para abrir a caixa de dialogo para seleção do arquivo. Esta caixa de dialogo filtra somente arquivos com extensão SVG. Esta funcionalidade está disponibilizada para qualquer usuário e poderá ser utilizado somente após a seleção de uma estrutura.

No caso de uso UC 06 – Cadastrar compartimento, o usuário irá cadastrar o compartimento da estrutura. A tela disponibilizada para o cadastro é aberta logo após a seleção do compartimento na estrutura importada no caso de uso UC 05 – Importar estrutura.

No caso de uso UC 07 – Registrar inspeção, o usuário irá registrar a inspeção desejada. Para acessar esta funcionalidade o usuário deverá clicar no botão Inspeção existente no componente de tabela existente no frame de cadastro de compartimento. O usuário informa o tipo de inspeção, data de inicio e fim, situação e status da inspeção, observações e por fim cadastra as pendências (UC 08 – Cadastrar pendência) da inspeção. Da forma que está construído o usuário só pode registrar uma inspeção se existe um compartimento cadastrado.

O usuário utiliza o UC 08 – Cadastrar pendência, para registrar as pendências de uma inspeção, esta funcionalidade esta localizada no mesmo frame disponibilizado no UC 07 – Registrar inspeção.

Para atender o caso UC 09 – Visualizar inspeção foi disponibilizado um componente de lista, onde cada inspeção registrada na base de dados é incluída nesta lista. Para o usuário visualizar a inspeção basta selecionar a inspeção na lista e ela será carregada nos campos de cadastro. A lista armazena em cada linha o código e a data inicial da inspeção.

No caso de uso UC 10 – Gerar gráfico/detalhe foi disponibilizado uma interface com filtros de data, funcionário, estrutura, projeto e compartimento e um botão para geração do gráfico. O usuário deve utilizar os filtros e o botão. A geração do gráfico é feita com a chamada de métodos do Framework JFreeChart.

No caso de uso UC 11 – Gerar relatório de informações da inspeção foi disponibilizado uma interface com filtros de data, funcionário, estrutura, projeto e compartimento e um botão para geração do relatório. O usuário deve utilizar os filtros e o botão. A geração do relatório é simples e o retorno da função é um arquivo com extensão .TXT. A saída do relatório foi feito em .TXT, pois não há tempo hábil para o estudo de uma ferramenta de geração de relatórios.

A seguir são demonstrados os detalhamentos dos três principais casos de uso identificado no diagrama de casos de uso da Figura 9. O detalhamento dos demais casos encontra-se no Apêndice B.

No Quadro 1 é demonstrado o detalhamento da importação da estrutura. Quando é mencionada a palavra usuário, considerar os atores Gestor e Funcionário.

UC05 - Importar estrutura.	
Pré-condição	01) UC04 – Cadastrar estrutura
Cenário principal	01) O usuário abre um projeto 02) O usuário abre uma estrutura 03) O usuário clica no botão de abertura de arquivo 04) O usuário seleciona o arquivo desejado 05) O sistema faz a leitura do arquivo 06) O sistema faz a montagem da lista dos arquivos que serão renderizados 07) O sistema renderiza os objetos extraídos do arquivo 08) O usuário clica no menu salvar estrutura
Cenário exceção 1	Nos passos 05 caso retorne uma exceção: O sistema emite uma mensagem para o usuário com o texto da exceção tratada
Pós-condição	O sistema mostra a estrutura gerada através de diretivas da API OpenGL

Quadro 1 - Caso de Uso 05 - importar estrutura

No Quadro 2 é demonstrado o detalhamento do cadastro de compartimento.

UC06 - Cadastrar compartimento.	
Pré-condição	01) UC05 – Importar estrutura
Cenário principal	01) O usuário abre um projeto 02) O usuário abre uma estrutura 03) O usuário seleciona o compartimento na estrutura renderizada, com a função de dois cliques do mouse 04) O sistema abre a tela de cadastro de compartimento 05) O sistema verifica se o compartimento já está cadastrado
Cenário alternativo 1	No passo 05, se estiver cadastrado: 05.1)O sistema seleciona o registro do compartimento no componente de tabela da janela. 05.2)O sistema desabilita os campos de cadastro.
Cenário alternativo 2	No passo 05, se não estiver cadastrado: 05.1)O sistema habilita os campos de cadastro 05.2)O usuário informa os dados do compartimento e clica em cadastrar 05.3)O sistema verifica se as informações estão válidas 05.4)O sistema atualiza o compartimento na tabela do banco de dados 05.6)O sistema atualiza o componente de tabela na janela 05.7)O sistema atualiza o frame renderizado.
Cenário alternativo 3	No passo 04, o usuário escolhe pesquisar um registro: 04.1)O usuário informa o código ou nome da estrutura 04.2)O usuário clica no botão pesquisar 04.3)O sistema localiza a estrutura e seleciona a linha da tabela correspondente ao registro pesquisado
Cenário alternativo 4	No passo 05, o usuário escolhe excluir um registro: 05.1)O usuário informa o código ou nome da estrutura 05.2)O usuário clica no botão excluir 05.3)O sistema verifica as informações de integridade 05.4)O sistema efetua a remoção do compartimento da base de dados 05.5)O sistema atualiza a lista de compartimentos 05.6)O sistema atualiza o frame renderizado
Cenário exceção 1	Nos passos 05.3 caso retorne uma exceção: O sistema emite uma mensagem para o usuário com o texto da exceção tratada
Pós-condição	O sistema mostra a informação cadastrada na tabela ou seleciona o registro

Quadro 2 - Caso de Uso 06 - cadastrar compartimento

No Quadro 3 é demonstrado o detalhamento do registro de inspeção.

UC07 - Registrar inspeção.	
Pré-condição	01) UC06 – Cadastrar compartimento
Cenário principal	01) O usuário abre um projeto 02) O usuário abre uma estrutura 03) O usuário seleciona um compartimento cadastrado 04) O usuário clica no botão Inspeção na tela cadastro de compartimento 05) O sistema abre a tela de registro de inspeção 06) O usuário seleciona o tipo de inspeção 07) O sistema carrega a lista de inspeções já registradas 08) O sistema habilita os campos de cadastro 09) O sistema carrega o script cadastrado no tipo de inspeção para o campo observação 10) O usuário informa os dados da nova inspeção 11) O usuário clica no botão gravar 12) O sistema verifica se as informações estão válidas 13) O sistema insere o registro da inspeção na tabela do banco de dados 14) O sistema insere a inspeção no componente de lista para ficar disponível para o usuário 15) O sistema informa o registro ao usuário
Cenário alternativo 1	No passo 09, se o usuário desejar excluir uma inspeção: 09.1)O usuário seleciona a inspeção no componente de lista 09.2)O sistema carrega as informações da inspeção nos respectivos campos 09.3)O usuário clica no botão de exclusão 09.4)O sistema exclui o registro da tabela do banco de dados 09.5)O sistema exclui o registro do componente de lista 09.6)O sistema informa a exclusão ao usuário
Cenário alternativo 2	No passo 15, se o usuário clicar no botão nova inspeção: 15.1)O usuário aciona o botão 15.2)O sistema limpa os campos da tela 15.3)O usuário retorna ao passo 10
Cenário exceção 1	No passo 12, se retornar exceção: 12.1)O sistema informa ao usuário a mensagem de critica
Pós-condição	A inspeção estará cadastrada ou excluída da base de dados

Quadro 3 - Caso de Uso 07 - registrar inspeção

3.2.2 Diagrama de classes

A apresentação dos diagramas de classes foi dividida em três partes, sendo que na primeira parte é apresentado o diagrama das classes de interface do sistema com o banco de dados MySQL, na segunda parte é apresentado o diagrama das classes utilizadas para a renderização da estrutura ou plano de capacidade da embarcação e na terceira e última parte é exibido o diagrama das classes de registro e controle das inspeções realizadas. Para melhor

entendimento, nas próximas seções será feito uma explanação de cada diagrama.

3.2.2.1 Diagrama de classes - interface com banco de dados

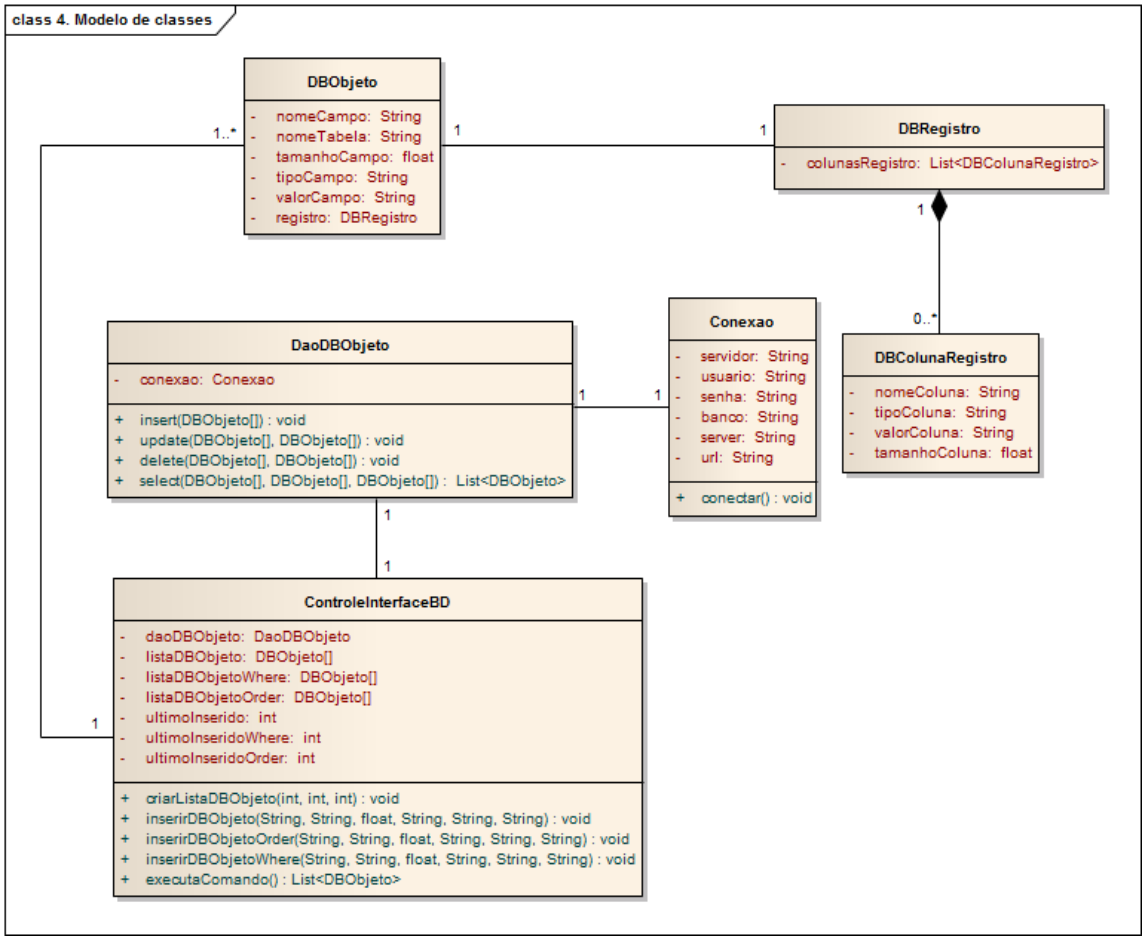


Figura 10 - Diagrama de classes - interface com banco de dados

Segue o detalhamento das classes apresentadas no diagrama da Figura 10, que são:

a) classe `ControleInterfaceBD`: responsável por armazenar as colunas dos comandos DML e por executar o comando conforme operação escolhida. A classe utiliza vetores de tamanhos fixos para facilitar a varredura e melhorar o desempenho no momento que a classe `DaoDBbjeto` cria os comandos DML;

b) classe `DaoDBbjeto`: responsável pela conexão com o banco de dados e por criar os comandos DML. Metodos `insert, update, delete e select,` são criados conforme os parâmetros passados;

c) classe `Conexão`: responsável por armazenar as informações da conexão com o banco de dados MySQL, armazena os atributos `servidor, usuário, senha,`

banco, server e url. Possui o método conectar que faz a chamada do DriverManager.getConnection passando os atributos como parâmetro;

d) classe DBObjeto: responsável por armazenar as informações dos comandos DML que serão criados. Os comandos são criados com base no atributo operação, respeitando os valores dos atributos nomeCampo, nomeTabela, tamanhoCampo, tipoCampo e valorCampo. O atributo registro* é utilizado apenas no comando select, onde armazena cada registro retornado do comando neste objeto;

e) classe DBRegistro: utilizada no comando select e responsável por armazenar as colunas de cada registro retornado no comando. O atributo principal é uma lista de objetos da classe DBColunaRegistro;

f) classe DBColunaRegistro: responsável por compor o objeto que armazena valores de cada coluna após a execução de comando select. É composta pelos atributos nomeColuna, tipoColuna, valorColuna e tamanhoColuna.

3.2.2.2 Diagrama de classes - renderização da estrutura da embarcação

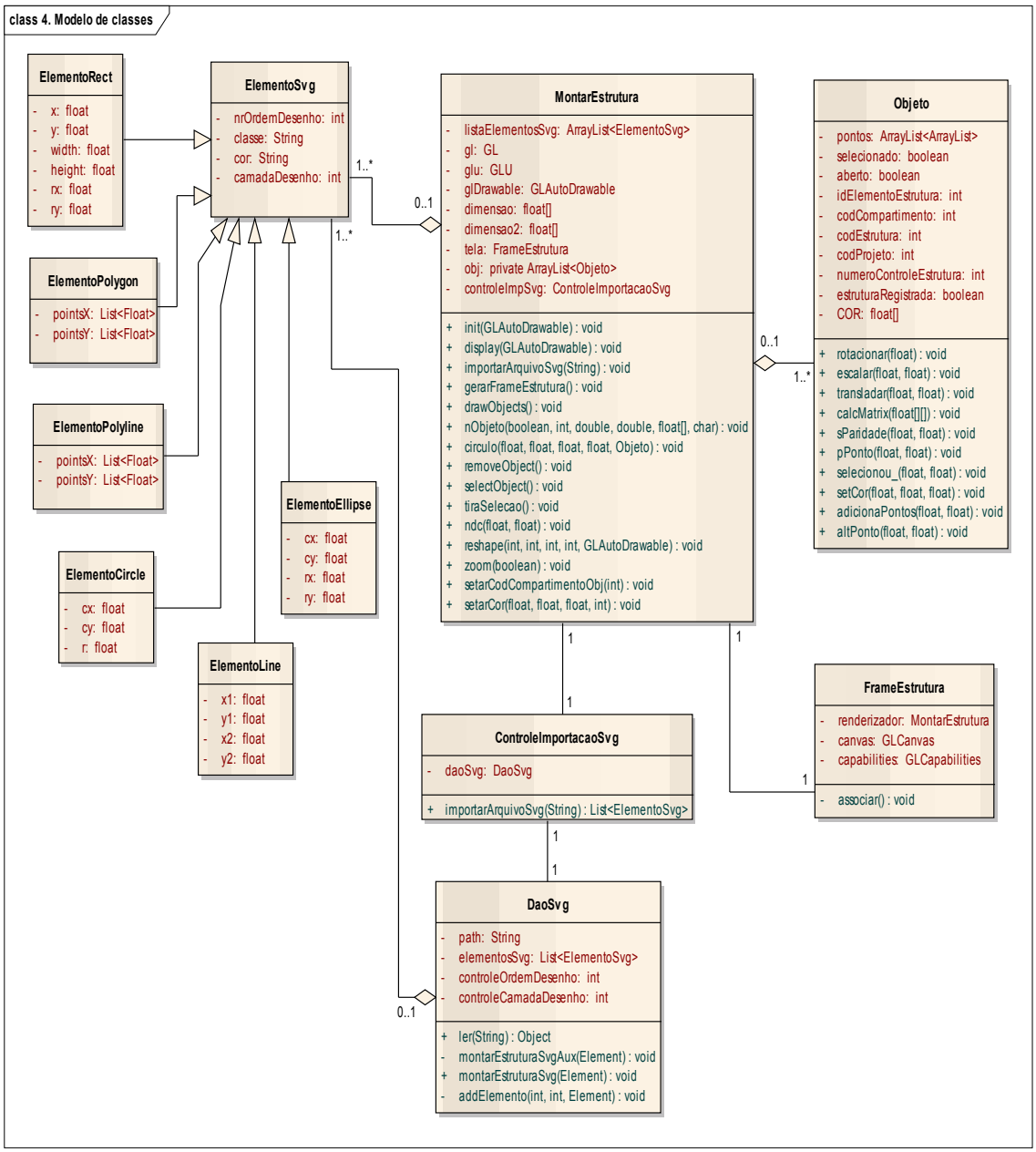


Figura 11 - Diagrama de classes - renderização

Segue o detalhamento das classes apresentadas no diagrama da Figura 11, que são:

a) classe `ElementoSvg`: classe pai, responsável por armazenar informações em comum aos objetos extraídos na leitura do arquivo SVG;

b) `ElementoRect`, `ElementoPolygon`, `ElementoPolyline`, `ElementoCircle`, `ElementoLine`, `ElementoEllipse` são classes filhas da classe `ElementoSvg`. Elas são responsáveis por armazenar informações dos tipos de elementos extraídos

na leitura do arquivo SVG. Cada classe tem seus atributos de acordo com o tipo do elemento;

c) classe `DaoSvg`: classe responsável por fazer a leitura do arquivo de extensão .SVG e montar a lista de elementos identificados no arquivo. Utiliza métodos recursivos e o *framework* jDom para leitura do arquivo;
d) classe `ControleImportacaoSvg`: classe de controle responsável pela execução do método de leitura do arquivo;
e) classe `MontarEstrutura`: classe responsável por todo o controle e renderização dos objetos extraídos do arquivo. É implementada com base nas diretivas da API OpenGL.
f) classe `Objeto`: classe responsável por armazenar dados do objeto que será renderizado no frame principal;
g) classe `FrameEstrutura`: classe de janela principal, responsável por armazenar os menus do programa e o frame contendo a estrutura gerada da embarcação.

3.2.2.3 Diagrama de classes - registro de inspeções

No diagrama de classes da Figura 12, tem-se o cenário das classes de controle utilizando a classe de interface com o banco de dados para inserir, consultar, excluir ou atualizar os registros nas tabelas do banco de dados.

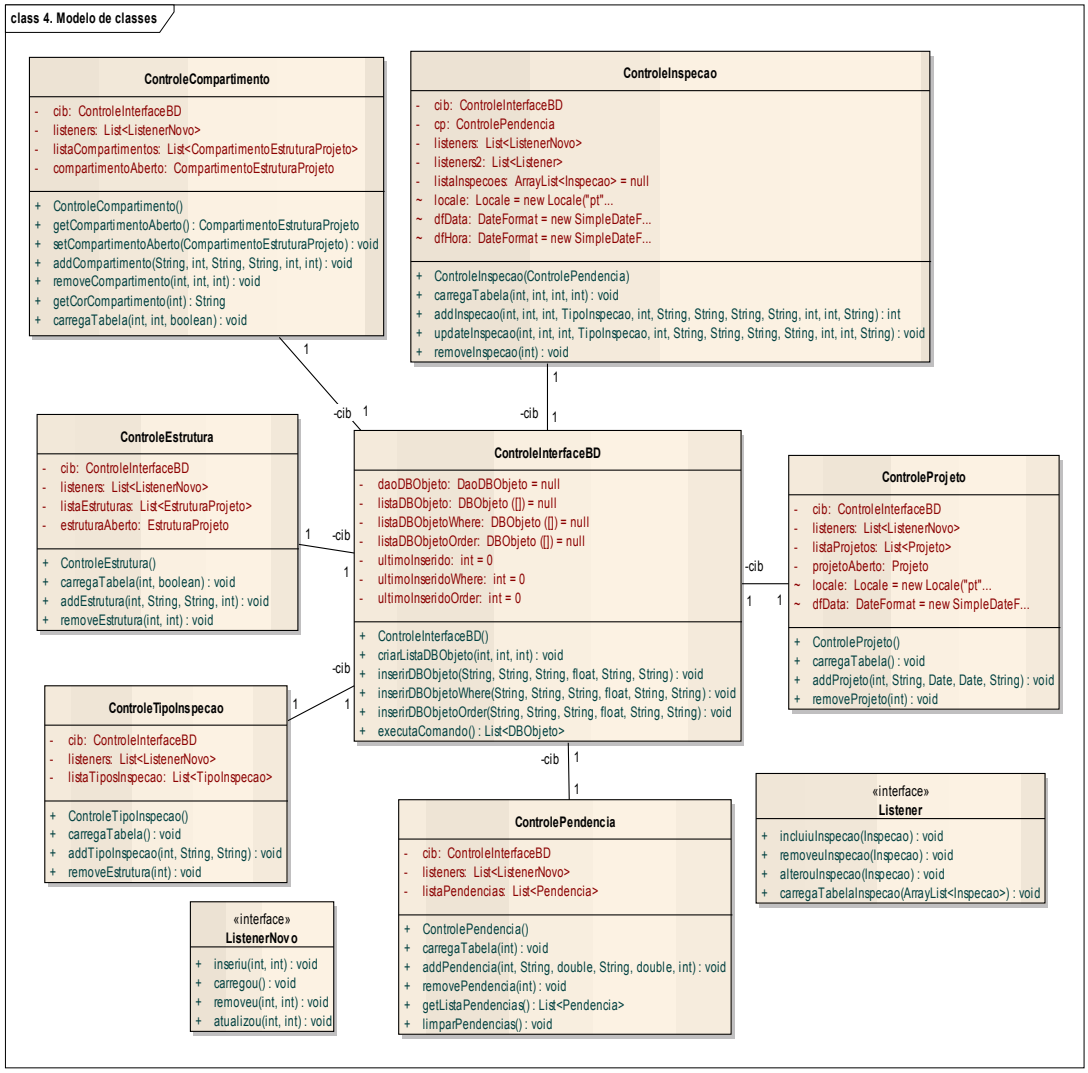


Figura 12 - Diagrama de classes - registro de inspeções

As classes de controle controlam as listas de objetos que são renderizados nos componentes de tabela das janelas de cadastro e consulta. Estas classes também executam os métodos conforme as funções solicitadas pelos usuários nas classes de janela do programa. Todas as classes desse diagrama têm praticamente as mesmas funções, alternando apenas o objeto manipulado.

As interfaces `Listener` e `ListenerNovo` servem para atualizar a janela ou os *models* dos componentes utilizados.

A Figura 13 apresenta o diagrama das classes responsáveis por modelar a hierarquia projeto, estrutura, compartimento e inspeção.

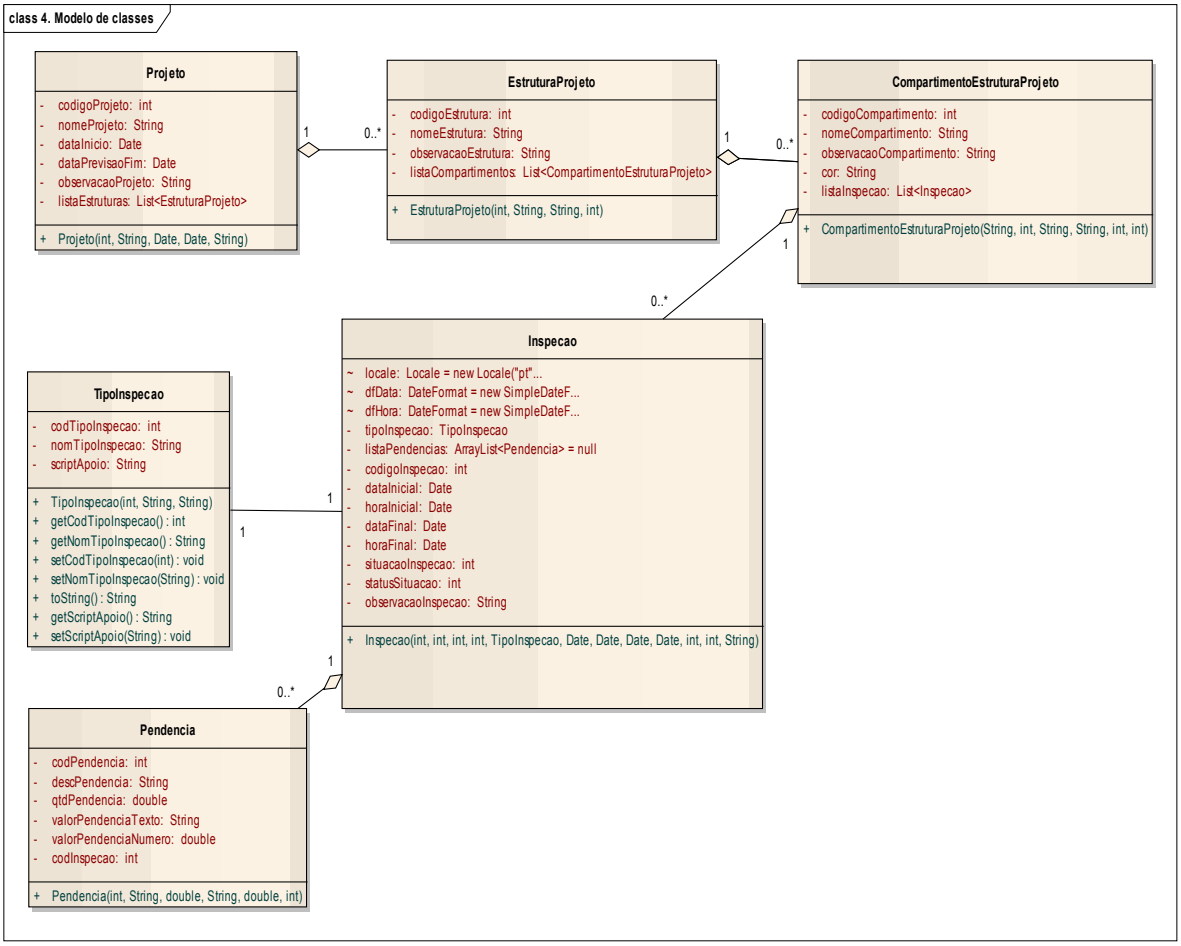


Figura 13 - Diagrama de classes - registro de inspeções

Segue o detalhamento das classes apresentadas no diagrama da Figura 13:

a) classe `Projeto`: responsável por armazenar dados do projeto e também possui um atributo para lista de estruturas de cada projeto;

b) classe `EstruturaProjeto`: responsável por armazenar dados da estrutura e também possui um atributo para lista de compartimentos;

c) classe `CompartimentoEstruturaProjeto`: responsável por armazenar dados do compartimento e também possui um atributo de lista de inspeções;

d) classe `Inspecao`: responsável por armazenar dados da inspeção e também possui um atributo de lista para armazenar as pendências da inspeção;

e) classe `TipoInspecao`: responsável por indicar o tipo de inspeção associado a cada inspeção. Armazena as informações de tipo de inspeção e nesta classe é importante ressaltar a existência de um atributo tipo String que armazena um script de execução da inspeção, serve como um passo a passo para o funcionário executar a inspeção. Este atributo é carregado no momento em que o usuário está cadastrando

uma nova inspeção;

f) classe `Pendencia`: responsável por armazena os dados das pendências das inspeções.

3.2.3 Diagrama de sequência

Os diagramas de sequência apresentados a seguir exibem o fluxo de utilização do sistema para completar um ciclo em que é incluído o cadastro de um novo projeto, o cadastro de uma nova estrutura, a importação da estrutura, o cadastro do compartimento selecionado e o registro da inspeção desejada.

Na Figura 14 é exibido o diagrama de sequência onde o usuário solicita a abertura de uma estrutura previamente cadastrada. O principal método desse fluxo é o método `drawObjects` que é responsável pela renderização do desenho. Ele está localizado na classe de controle `MontarEstrutura` que recebe como parâmetro a lista de objetos a serem renderizados. A classe `Objeto` é responsável por fornecer os valores de coordenadas dos objetos para renderização, estes valores são capturados com a leitura dos dados do arquivo no processo exibido na Figura 13. O resultado final do ciclo é a estrutura da embarcação gerada no frame da janela principal. O método `associar` é responsável por integrar a classe de controle `MontarEstrutura` como renderizador do frame principal localizado na classe `FrameEstrutura`, este processo é feito com a implementação da interface `GLEventListener` por parte da classe `MontarEstrutura`.

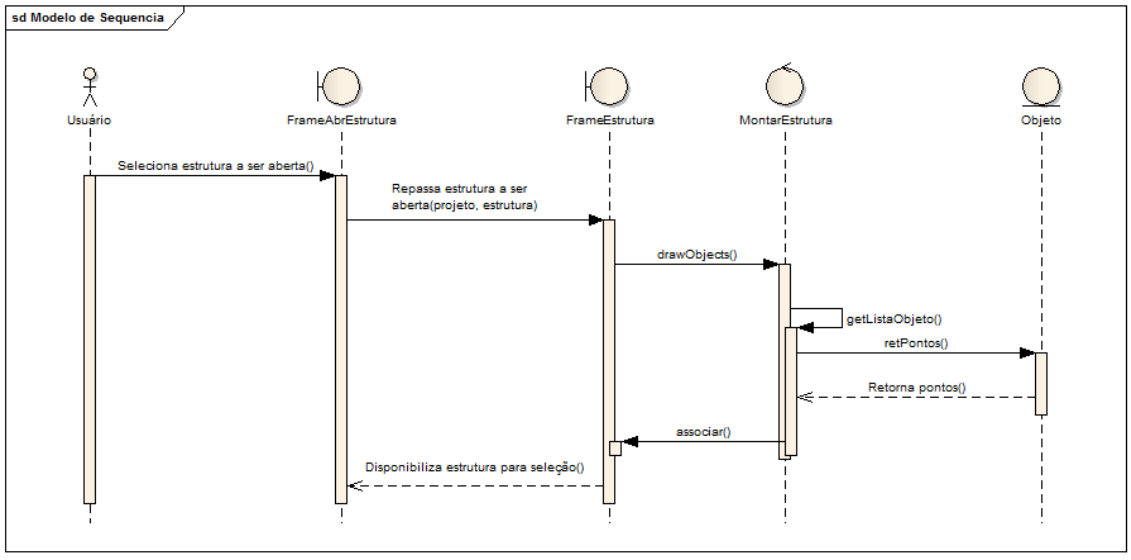


Figura 14 - Diagrama de sequência – abrir estrutura

A Figura 15 ilustra a troca de mensagens entre os objetos da tela principal, onde está contido o frame responsável por mostrar a estrutura da embarcação e as classes de tela e de controle responsáveis por registrar o compartimento e registrar a inspeção.

Na primeira passagem a troca de mensagens inicia-se com o clique do mouse pelo usuário e termina com a tela de cadastro de compartimento, disponibilizando o compartimento para inspeção. Entre estas etapas o usuário informa os dados do compartimento para o sistema que se encarrega de atualizar a base de dados. O registro de inspeção inicia somente após o cadastro do compartimento estar completo, lembrando que podem existir várias inspeções para apenas um compartimento.

No componente de tabela existente na tela de cadastro de compartimento foi criado um botão, responsável por abrir a tela de registro de inspeções. Este botão é utilizado na próxima passagem do diagrama onde o usuário deseja registrar uma inspeção para determinado compartimento. O usuário deve inserir os dados nos campos de cadastro e então gravar, o sistema se encarregará de fazer a criação do objeto de inspeção e disponibilizar para a classe `ControleInspecao` inserir o registro na base de dados e também na lista de controle. O sistema avisará ao usuário se ocorreu algum erro ou se gravou os dados na base corretamente. Nesse diagrama também foi especificado o cenário de exclusão de um compartimento. A exclusão ocorre com a seleção do compartimento na tela de cadastro e com o clique no botão de exclusão existente na janela, este botão faz a chamada do método excluir na classe `ControleCompartimento`.

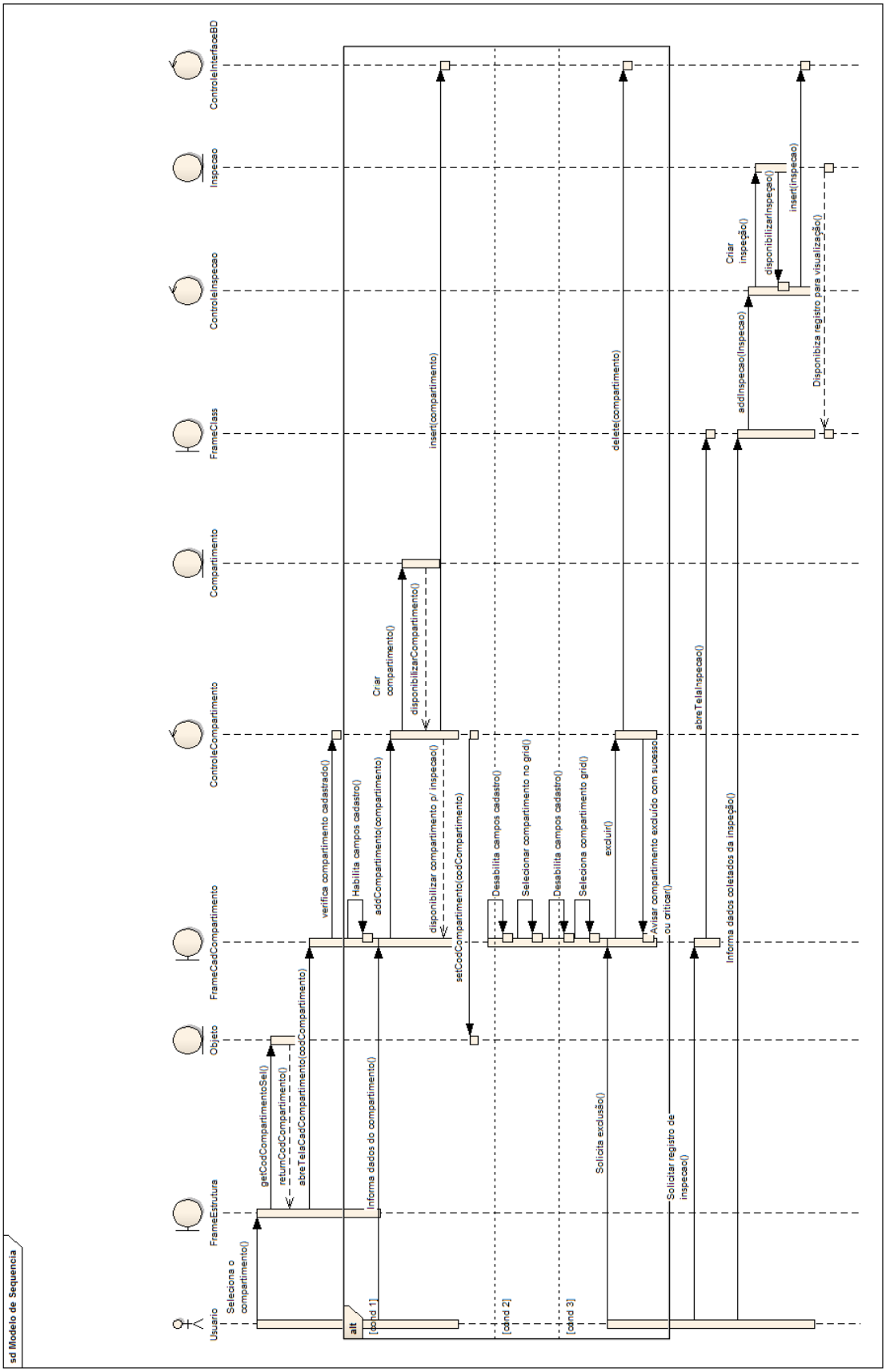


Figura 15 - Diagrama de sequência – registro de inspeção

A Figura 16 exibe o diagrama com a troca de mensagens entre os objetos responsáveis por fazer a leitura do arquivo e também os objetos responsáveis por renderizar as estruturas que compõe o desenho extraído, destacando que é possível importar apenas um desenho por estrutura devido ao sobre-carregamento de informações que dificultam a visualização dos compartimentos. Neste diagrama existem duas condições, uma para quando a estrutura já foi gravada na base de dados e outra para a estrutura que ainda não foi gravada. Para a primeira condição o usuário recebe uma mensagem de critica informando que já foi registrado um desenho para aquela estrutura. Para a segunda condição, o usuário segue o fluxo selecionando o arquivo a ser importado, após isso o sistema se encarregará de processar a solicitação, utilizando a chamada dos métodos recursivos `MontarEstruturaSvgAux` e `MontarEstruturaSvg` para leitura e captura dos atributos do arquivo que segue o padrão XML. O retorno deste método é a lista de objetos a serem renderizados no método `drawObjects`. Após esse processo ao sistema disponibiliza para o usuário a estrutura para seleção dos compartimentos.

Em seguida o usuário opta por salvar ou não os dados da estrutura na base de dados. Este processo implica na chamada do método `gravarEstruturaBD` da classe `ControleBDEstrutura`, onde para cada elemento existente na lista retornada do método recursivo citado acima, é criado um comando para inserir o mesmo na base de dados (vide seção 3.2.4).

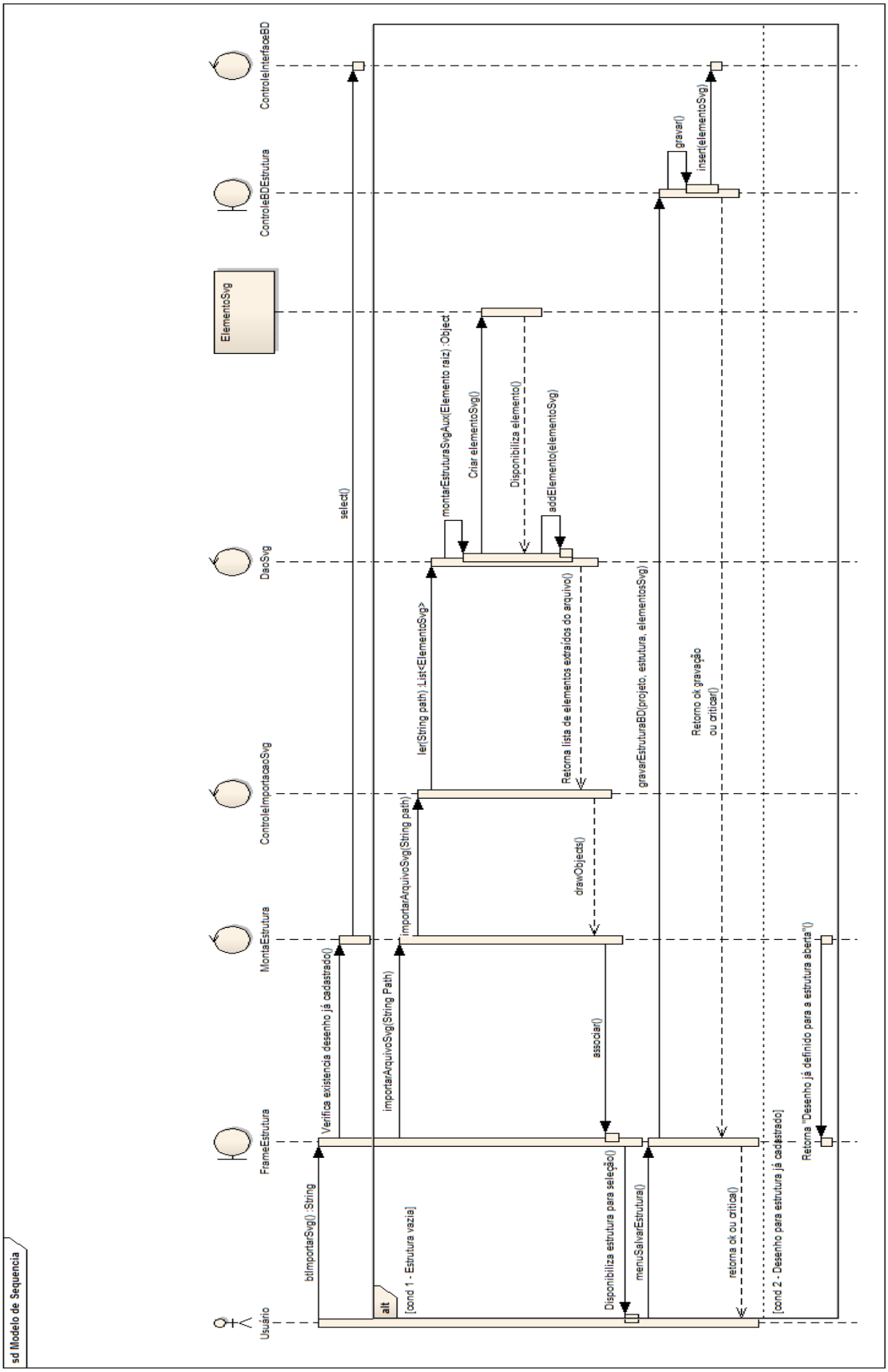


Figura 16 - Diagrama de sequência – importar estrutura

3.2.4 Modelo de Entidade Relacionamento

O diagrama entidade relacionamento da Figura 17 foi desenvolvido com apoio do software *MySQL Workbench* (MYSQL, 2012). Ele descreve de maneira conceitual os dados a serem utilizados no software.

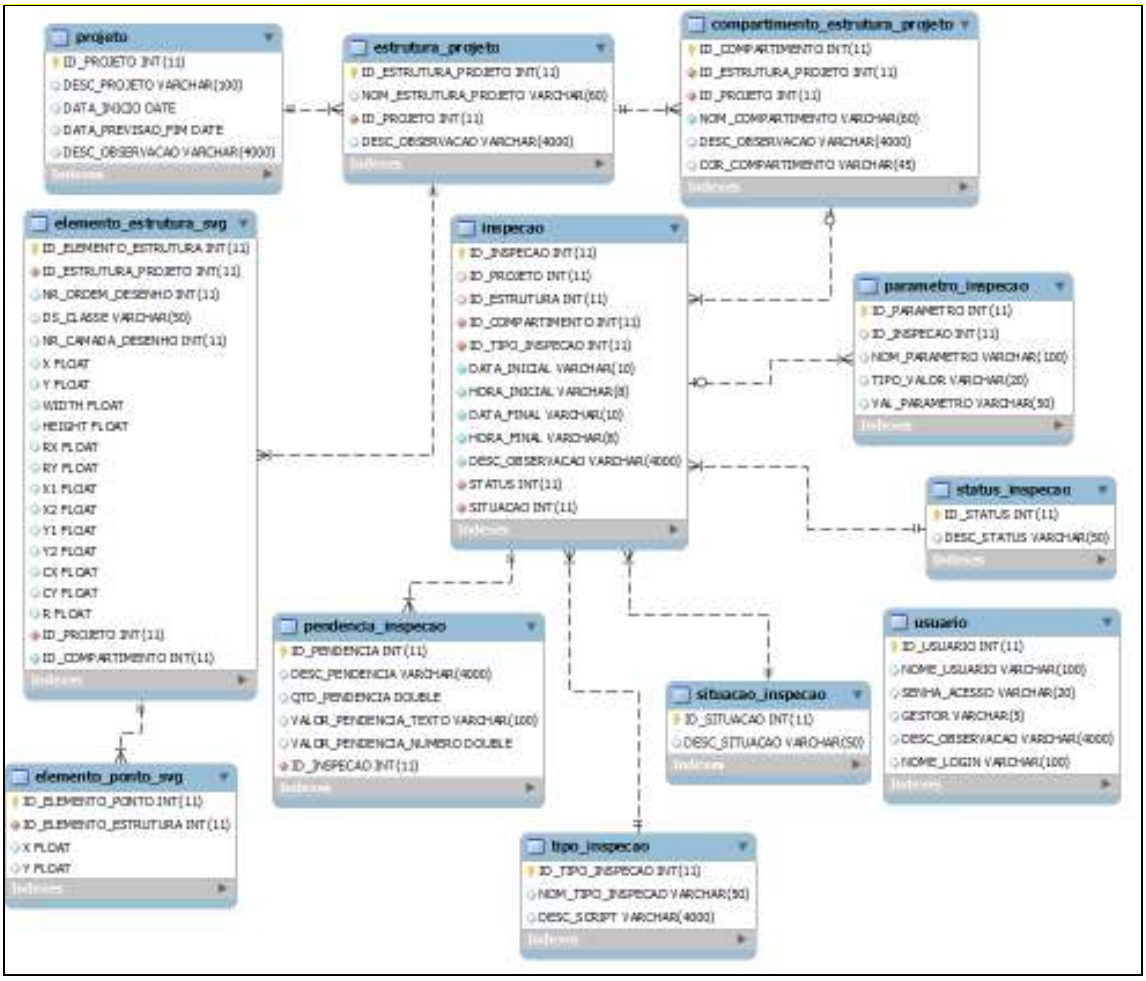


Figura 17 - Modelo entidade relacionamento

A seguir é feito o detalhamento da função de cada tabela utilizada para armazenamento dos dados.

a) tabela `projeto`: local onde são armazenados os dados dos projetos cadastrados no sistema;
b) tabela `estrutura_projeto`: local onde são armazenados os dados das estruturas de cada projeto;
c) tabela `compartimento_estrutura_projeto`: local onde são armazenados dados dos compartimentos de cada estrutura dos projetos;

d) tabela `usuario`: local onde são armazenados os dados dos usuários cadastrados no sistema;
e) tabela `elemento_estrutura_svg`: local onde são armazenados os dados dos objetos extraídos do arquivo e que serão renderizados;
f) tabela `elemento_ponto_svg`: local de armazenamento dos pontos dos atributos tipo arraylist das classes `ElementoPolygon` e `ElementoPolyline`;
g) tabela `inspecao`: local onde são armazenados os dados de cada inspeção registrada no sistema;
h) tabela `pendencia_inspecao`: local onde são armazenados os dados de cada pendência de cada inspeção;
i) tabela `tipo_inspecao`: local onde são armazenados os dados dos tipos de inspeção existentes;
j) tabela `parametro_inspecao`: local onde são armazenados dados dos parâmetros de inspeção.
k) Tabela `status_inspecao`: local onde são armazenados os dados dos status de inspeção existentes;
l) Tabela `situação_inspeção`: local onde são armazenados os dados das situações de inspeções existentes.

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas as técnicas, métodos e recursos utilizados para o desenvolvimento do sistema proposto.

3.3.1 Técnicas e ferramentas utilizadas

O sistema foi desenvolvido utilizando técnicas de orientação a objetos e o ambiente de desenvolvimento Eclipse Java *Enterprise Edition*. Para persistência dos dados foi utilizado o banco de dados MySQL na versão 5.1 e o ambiente *MySQL Workbench* para testes e execução dos comandos DDL e DML.

A linguagem de programação utilizada é o JAVA e as principais bibliotecas de apoio utilizadas são: jDom-1.1.3.jar, jFreeChart-1.0.13.jar e jogl.jar.

A API OpenGL é executada com apoio da biblioteca jogl.jar e de quatro DLLs, gluegen-rt.dll, jogl_awt.dll, jogl_cg.dll e jogl.dll.

3.3.2 Operacionalidade da implementação

Nesta seção são apresentadas as principais funcionalidades e interfaces do sistema desenvolvido. Serão exibidas na ordem de fluxo de utilização do sistema, onde o usuário conectado efetua o cadastro ou a abertura de um projeto já existente, logo em seguida efetua o cadastro da estrutura ou opta por abrir uma estrutura já definida para o projeto aberto.

O usuário é responsável por importar a estrutura, escolhendo um arquivo de extensão SVG que contém os pontos que definem o desenho da estrutura.

3.3.2.1 Importando a estrutura a partir de um arquivo SVG

Esta seção é composta por imagens mostrando o código fonte de alguns métodos das classes `DaoSvg` e `ControleImportacaoSvg`. Na linha 303 da Figura 18 é exibido o código fonte do botão responsável por fazer a chamada do método `importarArquivoSvg` que realiza a importação da estrutura do navio a partir de um arquivo SVG.

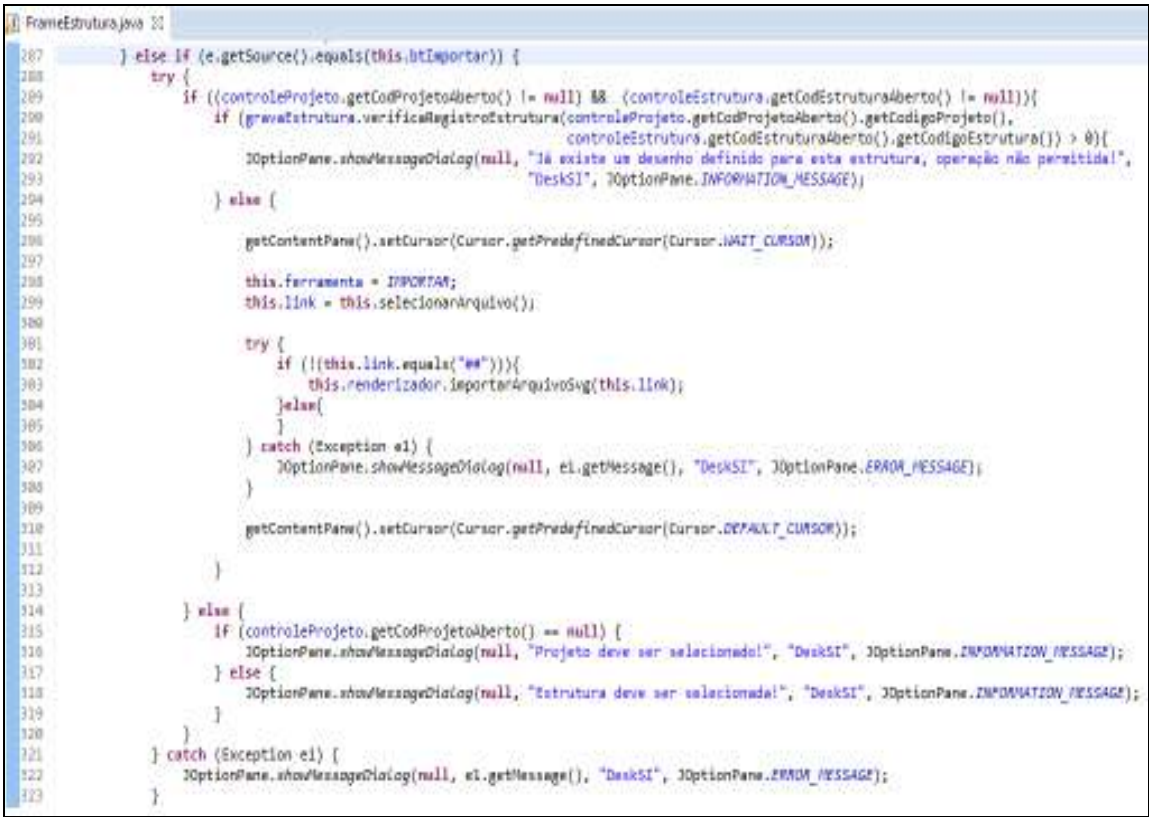


Figura 18 – Código fonte do botão de importação do arquivo

O método `selecionarArquivo`, na linha 299, faz a chamada da API do Windows responsável pela abertura da janela de seleção do arquivo no diretório desejado. O retorno desse método é passado para a classe de controle `MontarEstrutura`, neste caso o *handle* desta classe foi chamado de `renderizador` que por sua vez faz a chamada do método `importarArquivoSvg`.

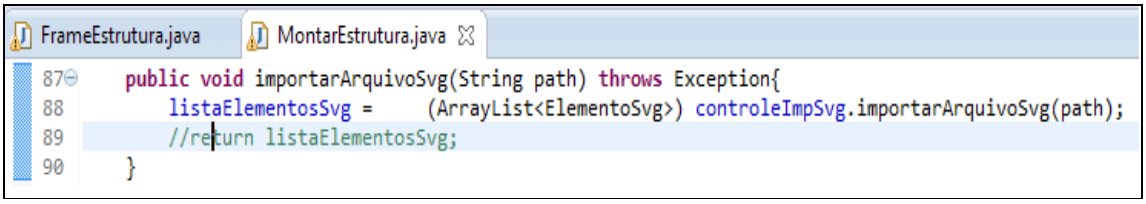


Figura 19 - Método `importarArquivoSvg`

A sequência de chamadas desta funcionalidade segue com o código fonte da Figura 20 onde são exibidos os três métodos responsáveis pela leitura e armazenamento dos elementos contidos no arquivo. Esta parte é feita de modo recursivo, conforme dito antes, com a utilização da classe `Document` que faz o mapeamento em memória da árvore *Document Object Model* (DOM) responsável pela representação do documento XML, da classe `SAXBuilder` que faz a compilação do documento XML e da classe `Element` utilizada para representar cada elemento contido na árvore. A utilização destas classes podem ser vistas entre as linhas 38 e 46 da Figura 20.

Os principais métodos utilizados destas classes são:

a) `getRootElement()` – responsável pela leitura do elemento raiz do arquivo;

b) `getChildren()` – responsável pela leitura do elemento filho de cada elemento contido na arvore;

c) `getName()` – responsável pela leitura do atributo *Name* de cada elemento;

d) `getAtributteValue()` – responsável pela leitura dos valores dos atributos passados como parâmetro.

FrameEstrutura.java | *DaoSvg.java

```
32    @Override
33    public Object ler(String p) throws Exception {
34        if (!p.equals(" ")) {
35            this.path = p;
36        }
37
38        Document doc = null;
39        SAXBuilder builder = null;
40
41        try{
42            File f = new File(this.path);
43            builder = new SAXBuilder();
44            doc = builder.build(f);
45
46            Element elementoEstrutura = doc.getRootElement();
47            montarEstruturaSvgAux(elementoEstrutura);
48        }catch(Exception e){
49            throw new Exception("Erro ao executar leitura do arquivo "+this.path+"!\n"+
50                                "Erro: "+e.getMessage());
51        }
52        return elementosSvg;
53    }
54
55    private void montarEstruturaSvgAux(Element elementoRaiz) throws Exception{
56        addElemento(elementoRaiz, this.controleOrdemDesenho, this.controleCamadaDesenho);
57        montarEstruturaSvg(elementoRaiz);
58    }
59
60    private void montarEstruturaSvg(Element elementoRaiz) throws Exception{
61        List<Element> lista = elementoRaiz.getChildren();
62        this.controleCamadaDesenho++;
63        for (Element e : lista) {
64            addElemento(e, this.controleOrdemDesenho, this.controleCamadaDesenho);
65            montarEstruturaSvg(e);
66        }
67        this.controleCamadaDesenho--;
68    }
```

Figura 20 - Código fonte do método recursivo de leitura do arquivo

A Figura 21 mostra o método responsável pelo armazenamento dos elementos de interesse ao sistema. É feito uma verificação para cada elemento encontrado no arquivo, sendo criado um novo objeto para seu respectivo tipo, por exemplo na linha 71 e 97 da Figura 21 são exibidas as consistências para elementos do tipo RECT e CIRCLE, logo após esta consistência, são capturados os dados de coordenadas de cada elemento para criação do objeto. Os elementos de interesse ao sistema são as classes filhas da classe `ElementoSvg` que estão especificadas no diagrama de classes da Figura 10 (seção 3.2.2.2).

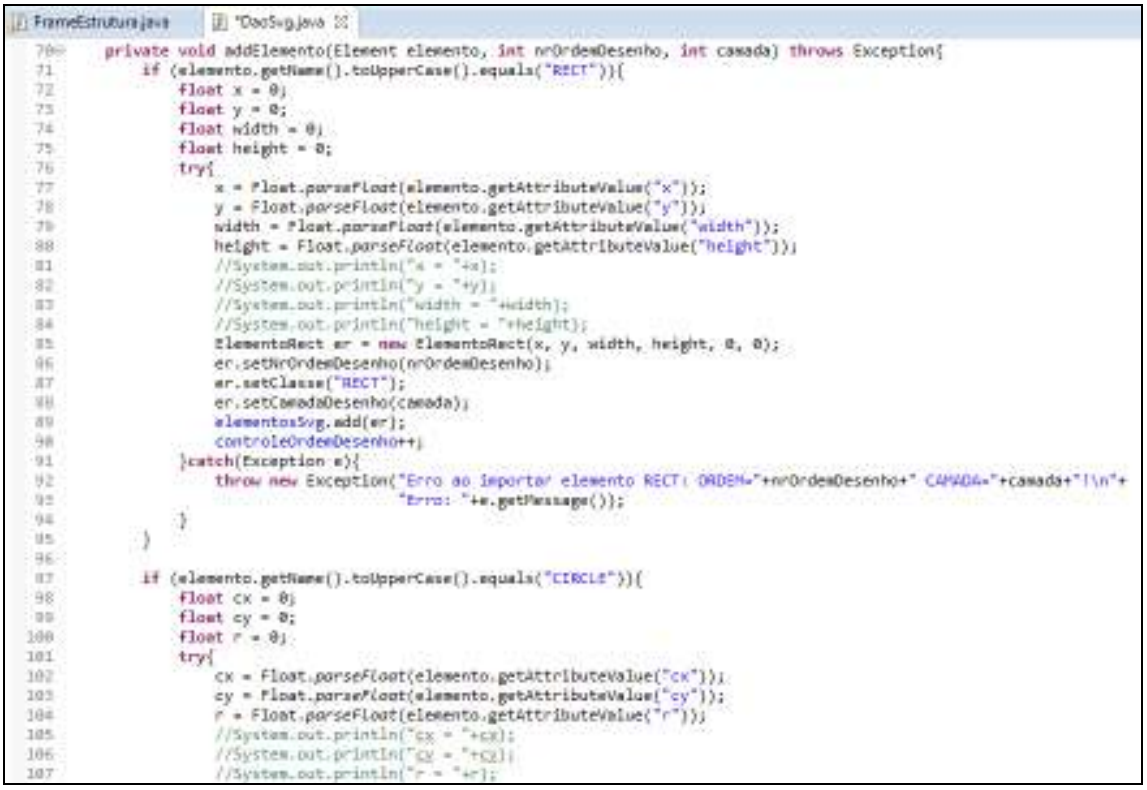


Figura 21 - Código fonte para armazenar os elementos encontrados

Após esta leitura é feito a renderização dos dados no frame de controle que será demonstrado na próxima seção.

3.3.2.2 Visualizando a estrutura e selecionando o compartimento

O processo de visualização da estrutura inicia-se com o término da leitura dos elementos do arquivo e com a chamada dos métodos `gerarFrameEstrutura` e `drawObject`.

O método `gerarFrameEstrutura` é responsável por fazer a leitura dos elementos encontrados na lista retornada dos métodos da seção anterior e fazer a distribuição dos dados aos objetos que serão renderizados. Observa-se na linha 121 da Figura 22 a criação de um novo objeto renderizável e a distribuição dos pontos de um elemento tipo retângulo para este objeto (linhas 122, 126, 130 e 134). O parâmetro 'P' passado ao método `nObjeto` (Figura 25) indica que é um objeto tipo polígono e que poderá ser aberto ou fechado (linha 137), dependendo da situação e das coordenadas extraídas do arquivo.

A distribuição dos dados aos objetos é feita com apoio dos métodos `adicionaPontos` (vide Figura 23) e ndc (vide Figura 24), sendo que o primeiro se encarrega de criar a lista de

pontos do objeto e atribuir os valores ao mesmo e o segundo faz a conversão dos pontos para coordenadas normalizadas de dispositivo.

FrameEstrutura.java *MontarEstrutura.java

```
92      public void gerarFrameEstrutura(){
93          // CONTROLAR COR
94          float red   = 0;
95          float green = 0;
96          float blue  = 0;
97
98          // CONTROLAR ABERTURA DA INSPECAO
99          int numControleEstrutura = 1;
100
101         for (ElementoSvg elementoSvg : listaElementosSvg) {
102             String cor = elementoSvg.getCor();
103             if (cor != null) {
104                 if (!cor.equals("") || !cor.equals("null")){
105                     String [] c = elementoSvg.getCor().split(",");
106                     red   = Float.parseFloat(c[0]);
107                     green = Float.parseFloat(c[1]);
108                     blue  = Float.parseFloat(c[2]);
109                 }
110             }
111
112             if (elementoSvg.getClasse().equals("RECT")){
113                 float x = ((ElementoRect)elementoSvg).getX();
114                 float y = ((ElementoRect)elementoSvg).getY();
115                 float width = ((ElementoRect)elementoSvg).getWidth();
116                 float height = ((ElementoRect)elementoSvg).getHeight();
117
118                 // RETÂNGULO 4 PONTOS
119                 // PRIMEIRO PONTO
120                 ult_ponto = ndc(x,y);
121                 nObjeto('P', null, 0, 0, numControleEstrutura, false);
122                 obj.get(obj.size()-1).adicionaPontos(ult_ponto[0], ult_ponto[1]);
123
124                 // SEGUNDO PONTO
125                 ult_ponto = ndc(x+width,y);
126                 obj.get(obj.size()-1).adicionaPontos(ult_ponto[0], ult_ponto[1]);
127
128                 // TERCEIRO PONTO
129                 ult_ponto = ndc(x+width,y+height);
130                 obj.get(obj.size()-1).adicionaPontos(ult_ponto[0], ult_ponto[1]);
131
132                 // QUARTO PONTO
133                 ult_ponto = ndc(x,y+height);
134                 obj.get(obj.size()-1).adicionaPontos(ult_ponto[0], ult_ponto[1]);
135
136                 // FECHA POLIGONO
137                 obj.get(obj.size() - 1).setAberto(false);
138
139                 obj.get(obj.size() - 1).setIdElementoEstrutura(elementoSvg.getIdElementoEstrutura());
140                 obj.get(obj.size() - 1).setCodCompartimento(elementoSvg.getCodCompartimento());
141                 obj.get(obj.size() - 1).setCodEstrutura(elementoSvg.getCodEstrutura());
142                 obj.get(obj.size() - 1).setCodProjeto(elementoSvg.getCodProjeto());
143
144                 if (cor != null) {
145                     if (!cor.equals("") || !cor.equals("null")){
146                         obj.get(obj.size() - 1).setCor(red, green, blue);
147                     }
148                 }
149
150                 numControleEstrutura ++;
151             }
```

Figura 22 - Distribuição de pontos dos objetos a serem renderizados

*MontarEstrutura.java | Objeto.java

```
34      public void adicionaPontos(float x, float y){
35          ArrayList<Float> pt = new ArrayList<Float>();
36          pt.add(x);
37          pt.add(y);
38          pontos.add(pt);
39          calculaBB(x, y);
40      }
```

Figura 23 - Método `adicionaPontos`

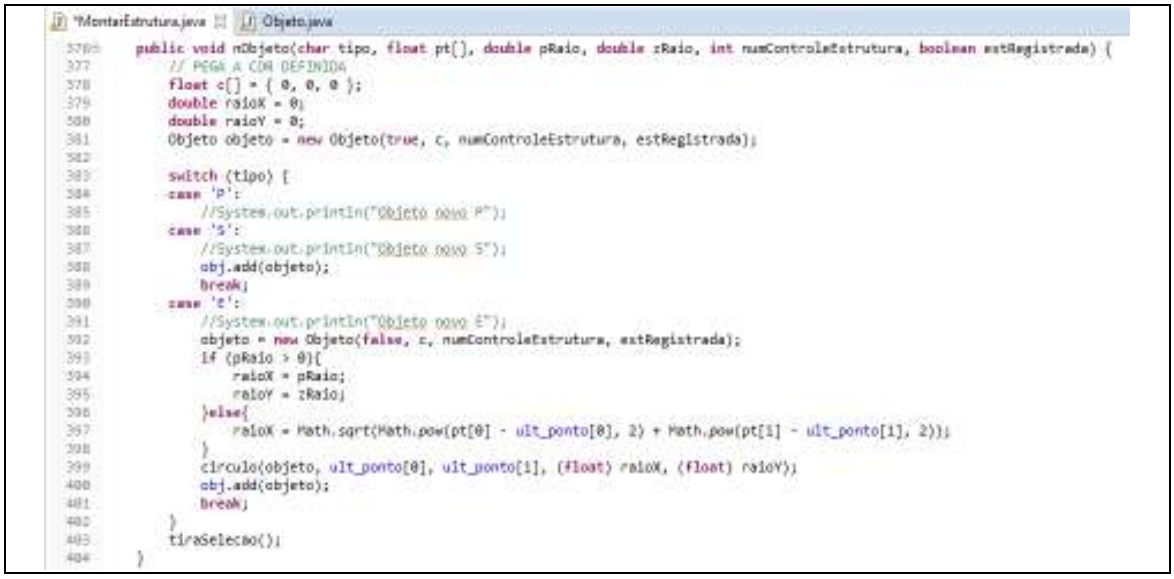
O método `ndc` exibido na Figura 24 é responsável por normalizar os valores dos pontos extraídos do arquivo. NDC é um sistema de coordenadas normalizadas que é intermediário entre o sistema de coordenadas do mundo real e o sistema de coordenadas de dispositivos. Este método foi criado para ajustar os valores das coordenadas extraídas do arquivo para valores referentes no plano cartesiano. O cálculo dos parâmetros x e y são feitos nas linhas 450 e 451, com base no plano-dimensão já iniciado com a instanciação da estrutura no método `init`.

*MontarEstrutura.java | Objeto.java

```
444     public float[] ndc(float ptx, float pty) {
445         // P INICIAL
446         float oix = 0, oiy = 0;
447         // P FINAL
448         float ofx = glDrawable.getWidth(), ofy = glDrawable.getHeight();
449         float[] pt = {
450                 ((ptx - oix) * ((dimensao[1] - dimensao[0]) / (ofx - oix))) + dimensao[0],
451                 ((pty - oiy) * ((dimensao[2] - dimensao[3]) / (ofy - oiy))) + dimensao[3] };
452         return pt;
453     }
```

Figura 24 - Método `ndc`

No método `nObjeto` da Figura 25 observa-se também como é criado os outros tipos de objetos do sistema, que podem ser Ellipses (linha 390), Círculos (linha 399) ou Splines (linha 386), sendo que esse último não é utilizado e não é suportado pela especificações dos arquivos SVG. O objeto criado é adicionado à lista de objetos com o comando na linha 400.

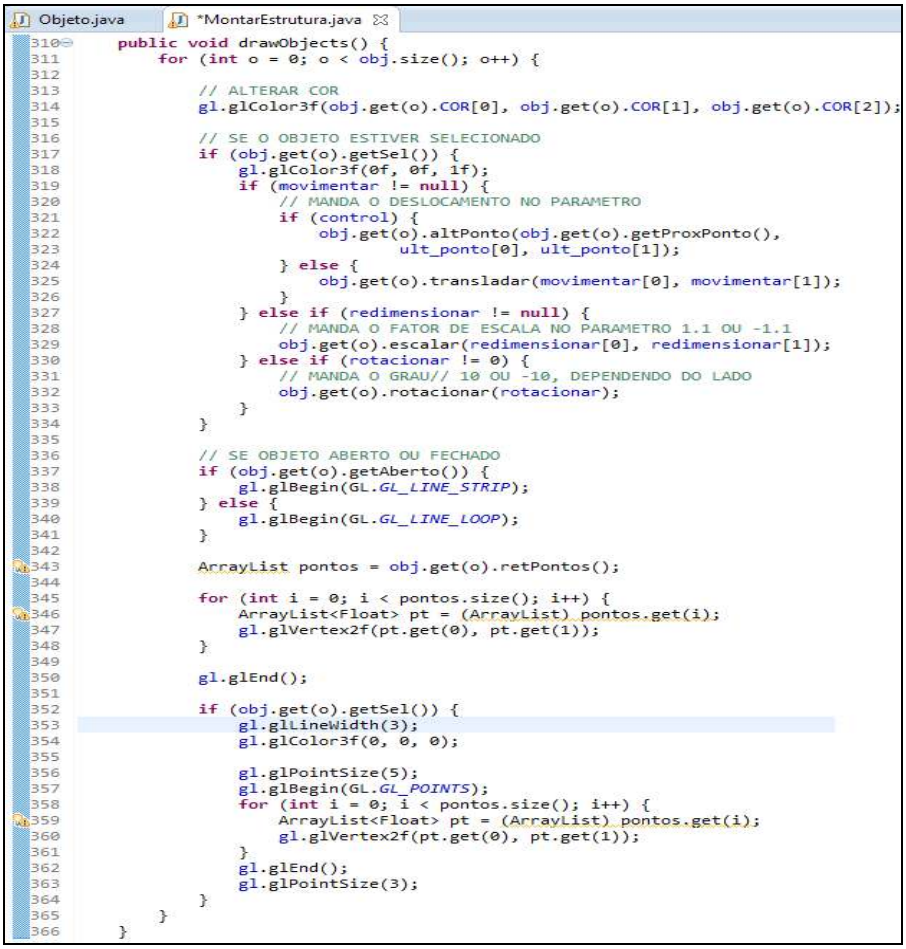


```
*MontarEstrutura.java   Objeto.java
    public void nObjeto(char tipo, float pt[], double pRaio, double zRaio, int numControleEstrutura, boolean estRegistrada) {
        // PEGA A COR DEFINIDA
        float c[] = { 0, 0, 0 };
        double raioX = 0;
        double raioY = 0;
        Objeto objeto = new Objeto(true, c, numControleEstrutura, estRegistrada);

        switch (tipo) {
        case 'p':
            //System.out.println("Objeto novo P");
        case 's':
            //System.out.println("Objeto novo S");
            obj.add(objeto);
            break;
        case 'c':
            //System.out.println("Objeto novo C");
            objeto = new Objeto(false, c, numControleEstrutura, estRegistrada);
            if (pRaio > 0){
                raioX = pRaio;
                raioY = zRaio;
            }else{
                raioX = Math.sqrt(Math.pow(pt[0] - ult_ponto[0], 2) + Math.pow(pt[1] - ult_ponto[1], 2));
            }
            circulo(objeto, ult_ponto[0], ult_ponto[1], (float) raioX, (float) raioY);
            obj.add(objeto);
            break;
        }
        tiraSelecao();
    }
```

Figura 25 - Método `nObjeto`

O método `drawObjects` da Figura 26 é responsável pela renderização dos objetos capturados nos passos anteriores. Para isto, usa a API OpenGL.



```
Objeto.java   *MontarEstrutura.java
    public void drawObjects() {
        for (int o = 0; o < obj.size(); o++) {

            // ALTERAR COR
            gl.glColor3f(obj.get(o).COR[0], obj.get(o).COR[1], obj.get(o).COR[2]);

            // SE O OBJETO ESTIVER SELECIONADO
            if (obj.get(o).getSel()) {
                gl.glColor3f(0f, 0f, 1f);
                if (movimentar != null) {
                    // MANDA O DESLOCAMENTO NO PARAMETRO
                    if (control) {
                        obj.get(o).altPonto(obj.get(o).getProxPonto(),
                                ult_ponto[0], ult_ponto[1]);
                    } else {
                        obj.get(o).transladar(movimentar[0], movimentar[1]);
                    }
                } else if (redimensionar != null) {
                    // MANDA O FATOR DE ESCALA NO PARAMETRO 1.1 OU -1.1
                    obj.get(o).escalar(redimensionar[0], redimensionar[1]);
                } else if (rotacionar != 0) {
                    // MANDA O GRAU// 10 OU -10, DEPENDENDO DO LADO
                    obj.get(o).rotacionar(rotacionar);
                }
            }

            // SE OBJETO ABERTO OU FECHADO
            if (obj.get(o).getAberto()) {
                gl.glBegin(GL.GL_LINE_STRIP);
            } else {
                gl.glBegin(GL.GL_LINE_LOOP);
            }

            ArrayList pontos = obj.get(o).retPontos();

            for (int i = 0; i < pontos.size(); i++) {
                ArrayList<Float> pt = (ArrayList) pontos.get(i);
                gl.glVertex2f(pt.get(0), pt.get(1));
            }

            gl.glEnd();

            if (obj.get(o).getSel()) {
                gl.glLineWidth(3);
                gl.glColor3f(0, 0, 0);

                gl.glPointSize(5);
                gl.glBegin(GL.GL_POINTS);
                for (int i = 0; i < pontos.size(); i++) {
                    ArrayList<Float> pt = (ArrayList) pontos.get(i);
                    gl.glVertex2f(pt.get(0), pt.get(1));
                }
                gl.glEnd();
                gl.glPointSize(3);
            }
        }
    }
```

Figura 26 - Método `drawObjects`

Para o correto funcionamento da OpenGL dentro da classe de controle `MontarEstrutura` foi necessário fazer com que ela implemente a interface `GLEventListener`, esta interface tem em sua definição quatro métodos. São eles:

a) `init`, chamado quando a OpenGL é iniciada. Responsável pelas configurações de métodos de chamada única. Esse método é exibido na Figura 27. Ele também define o plano-dimensão que é utilizado para manter os vértices e arestas dos objetos renderizados, o atributo responsável é exibido na linha 64;

*MontarEstrutura.java | Objeto.java

```
56      public void init(GLAutoDrawable drawable) {
57          glDrawable = drawable;
58          gl = drawable.getGL();
59          glu = new GLU();
60          glDrawable.setGL(new DebugGL(gl));
61
62          gl.glClearColor(1f, 1f, 1f, 1f);
63          gl.glLineWidth(3);
64          float dimensao3_nova[] = { -glDrawable.getWidth(),
65                                      glDrawable.getWidth(),
66                                     -glDrawable.getHeight(),
67                                      glDrawable.getHeight() };
68
69          dimensao = dimensao3_nova;
70          dimensao2 = dimensao3_nova;
71      }
```

Figura 27 - Método `init`

b) `display`, (Figura 28) responsável pela configuração de chamadas repetitivas de métodos de um loop, funciona como uma *thread,* nota-se a chamada do método `drawObjects` (linha 45) desta forma;

*MontarEstrutura.java | Objeto.java

```
28  public class MontarEstrutura implements GLEventListener,
29                                          KeyListener,
30                                          MouseListener,
31                                          MouseMotionListener {
32      private GL gl;
33      private GLU glu;
34      private GLAutoDrawable glDrawable;
35
36      public void display(GLAutoDrawable arg0) {
37          gl.glClear(GL.GL_COLOR_BUFFER_BIT);
38          gl.glMatrixMode(GL.GL_MODELVIEW);
39          gl.glLoadIdentity();
40
41          glu.gluOrtho2D(dimensao[0], dimensao[1], dimensao[2], dimensao[3]);
42
43          gl.glColor3f(0.9f, 0.9f, 0.9f);
44
45          drawObjects();
46
47          gl.glFlush();
48      }
```

Figura 28 - Método `display`

c) `reshape`, (Figura 29) utilizado no momento em que ocorre alteração do tamanho da janela. Na linha 472 é efetuado um cálculo para quando é alterada a largura da janela e na linha 478 o cálculo é feito para a altura da janela;

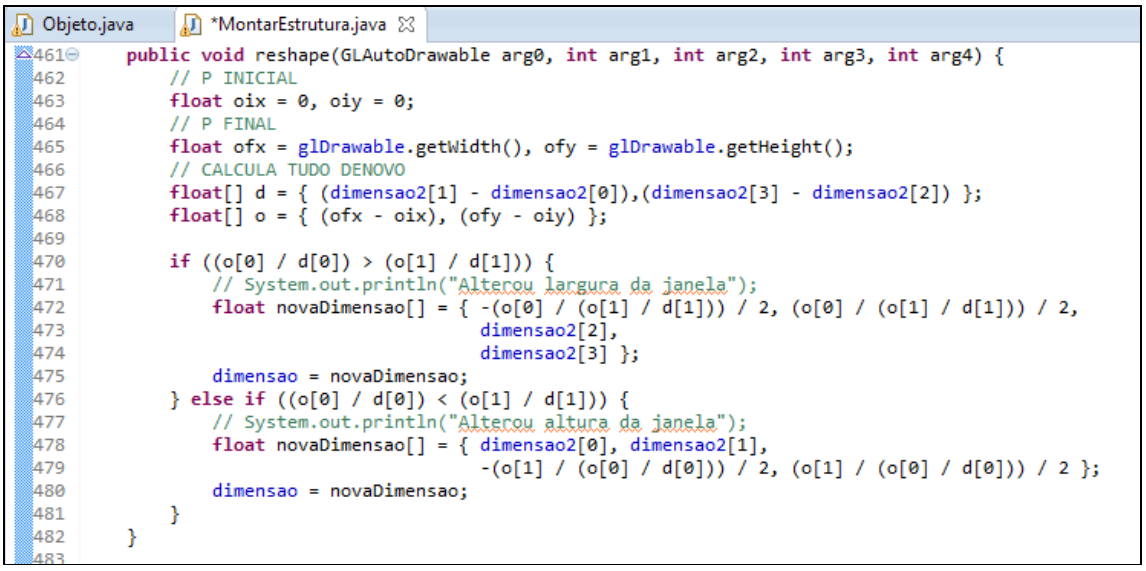


Figura 29 - Método `reshape`

d) `displayChanged`, não utilizado pelo sistema.

Além desses métodos foi necessário a criação dos atributos `gl`, `glu` e `glDrawable`. Estes atributos são instanciados no método `init` e são responsáveis pela ligação entre a classe de controle e a utilização das diretivas de renderização de cor, pontos e linhas, além de controlarem o espaço dimensional do canvas onde é exposto a estrutura.

A seguir são listadas a principais diretivas utilizadas nestes métodos:

a) `glClear`: responsável por limpar o buffer e por habilitar o buffer para escrita de cor;

b) `glMatrixMode`: avisa a OpenGL que todas as futuras alterações, tais como operações de escala, rotação e translação, irão afetar os modelos da cena, ou em outras palavras, o que é desenhado. A função `glLoadIdentity`; chamada em seguida, faz com que a matriz corrente seja inicializada com a matriz identidade (nenhuma transformação é acumulada);

c) `glLoadIdentity`: descrito no passo b;

d) `gluOrtho2D`: é usada para determinar que a projeção ortográfica (2D) será utilizada para exibir na tela a imagem 2D que está na janela de seleção definida através dos parâmetros passados para esta função. O protótipo desta função é: void `gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);` No sistema de coordenadas cartesianas, os

valores left e right especificam os limites mínimo e máximo no eixo X; analogamente, bottom e top especificam os limites mínimo e máximo no eixo Y;

e) `glColor3f`: determina a cor que será usada para o desenho (linhas e preenchimento). A seleção da cor é feita da mesma maneira que na função `glClearColor`, sendo que não é necessário especificar o componente alfa, cujo valor default é 1.0 (completamente opaco);

f) `glFlush`: responsável por limpar o buffer de execução e forçar a execução dos métodos da OpenGL novamente;

g) `glLineWidth`: responsável por definir a espessura da linha de desenho;

h) `glBegin/glEnd`: usado para desenhar os elementos a partir dos vértices especificados entre `glBegin` e `glEnd`;

i) `glPointSize`: responsável por definir o tamanho do ponto de desenho;

j) `glVertex2f`: diretiva responsável desenhar os vértices de acordo com os valores passados como parâmetro, neste caso a representação é de duas dimensões com parâmetros tipo float.

Para seleção do compartimento, foi utilizado o algoritmo de seleção por paridade, conforme mostra a Figura 30.

```
*Objeto.java   MontaEstrutura.java
1430    public boolean sParidade(float x, float y){
144         //CRIA ARRAY PARA GUARDAR OS PONTOS INTERNO
145         ArrayList<Float> inter = new ArrayList<Float>();
146         for (int i=0; i<pontos.size(); i++){
147             //BUSCA OS PARES DE PONTO, CASO FOR O ÚLTIMO FAZ PAR COM O PRIMEIRO
148             int ii = i + 1;
149             if (i == pontos.size()-1){ ii = 0; }
150             //MONTA OS PARES
151             float pt1[] = {((ArrayList<Float>)pontos.get(i)).get(0), ((ArrayList<Float>)pontos.get(i)).get(1)};
152             float pt2[] = {((ArrayList<Float>)pontos.get(ii)).get(0), ((ArrayList<Float>)pontos.get(ii)).get(1)};
153             float ti = (y - pt1[1]) / (pt2[1] - pt1[1]);
154
155             if (0 < ti && ti < 1){
156                 float xi = pt1[0] + (pt2[0] - pt1[0])*ti;
157                 inter.add(xi);
158             }
159             //ATE AQUI
160         }
161         int count = 0;
162         // SE O OBJETO FOR ABERTO
163         if (aberto == true){
164             for (int i=0; i<inter.size(); i++){
165                 if (Math.abs(inter.get(i) -x) < 1.0f){
166                     count = 1;
167                     break;
168                 }
169             }
170         //SE O OBJETO FOR FECHADO
171         }else{
172             for (int i=0; i<inter.size(); i++){
173                 if (inter.get(i) > x){
174                     count ++;
175                 }
176             }
177         }
178         //VERIFICA SE É IMPAR OU PAR
179         return count % 2 == 1;
180     }
```

Figura 30 - Método `sParidade`

O método `sParidade` é responsável por implementar a verificação do bit de paridade, esse método foi desenvolvido com intuito de prevenir a seleção de objetos indevidos, funciona da seguinte forma. Recebe como parâmetro o ponto x e y do clique do mouse, e efetua os cálculos entre todos os objetos contidos na lista de controle. O objetivo é verificar se o par de pontos capturado com o clique do mouse está dentro do objeto desejado. Para isso o algoritmo conta a quantidade de cruzamentos de arestas com esse segmento, se o número for impar, o clique está dentro do objeto, caso contrário estará fora (ITAPORANGASP, 2011). Segue o processo criado para quando o usuário clica para selecionar um compartimento na tela:

a) o método `selectObject` exibido na Figura 31 é responsável por varrer a lista de objetos (linha 423), realizando a chamada do método `selecionou_`.

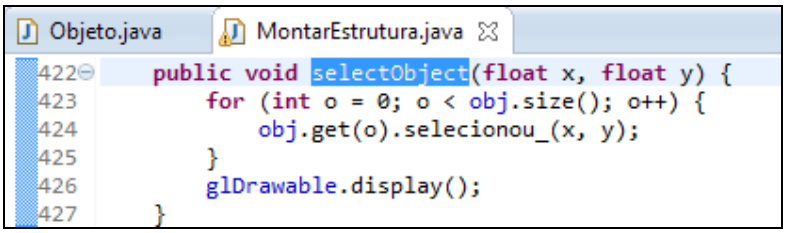


Figura 31 - Método `selectObject`

b) o método `selecionou_` exibido na Figura 32, faz a primeira verificação do ponto de clique com os pontos extremos do objeto (BoundBox). Caso a condição da linha 121 seja atendida uma variável é utilizada para marcar o objeto como selecionado, neste primeiro momento. Esta operação não garante que o objeto seja realmente selecionado, devido a BoundBox representar a caixa delimitadora de uma imagem, ou seja, ela é descrita por quatro números medidos a partir dos pontos mais extremos de cada lado, permitindo que ocorra situações em que, dentro da BoundBox exista partes que não pertençam ao objeto. Para tratar essa situação foi utilizado o método demonstrado no item c. O método `pPonto` que aparece na linha 130 da imagem abaixo, serve para buscar o vértice mais próximo ao ponto de clique.

Objeto.java | MontarEstrutura.java

```
120    public void selecionou_(float x, float y){
121        if (boundBox[0] < x & boundBox[1] > x & boundBox[2] < y & boundBox[3] > y){
122            selecionado = true;
123            pPonto(x,y);
124            selecionado = sParidade(x,y);
125        }else{
126            selecionado = false;
127        }
128    }
129
130    public void pPonto(float x, float y){
131        float p = Float.MAX_VALUE;
132        for (int i=0; i<pontos.size(); i++){
133            ArrayList<Float> pt = pontos.get(i);
134            float distancia = Math.abs(x - pt.get(0)) + Math.abs(y - pt.get(1));
135            if (distancia < p){
136                p = distancia;
137                proxPonto = i;
138            }
139        }
140    }
```

Figura 32 - Algorítmo de verificação de pontos de seleção

3.3.2.3 Adicionando nova inspeção

Nesta seção será demonstrado o funcionamento completo do sistema, desde a entrada até o registro da inspeção que é o objetivo principal do sistema. Para o registro da inspeção o usuário deve utilizar o sistema de acordo com o que está descrito na operacionalidade do trabalho e atender os cadastros pré-requisitos para então conseguir acesso a tela de registro de inspeções. Os dados inseridos no sistema são coletados pelo inspetor que seguirá o fluxo descrito nos próximos parágrafos para integrá-los ao sistema.

A Figura 33 exibe a tela de *login* do software e esta tela tem funcionalidade simples que controla a entrada de usuários no sistema conforme cadastrado previamente por um usuário gestor. A imagem contida na janela é substituída pelo logotipo da empresa.

Figura 33 - Tela de *login*

Na Figura 34 tem-se a tela principal do sistema onde são armazenados os menus e também o frame de controle da estrutura. Além desses componentes, são encontrados botões que acionam ferramentas para manipulação da imagem conforme desejado. A tela abre maximizada como padrão, visando melhorar o espaço para visualização da estrutura. Mesmo a janela abrindo maximizada, existe estruturas de grande porte onde é necessário utilizar a ferramenta de zoom para facilitar a leitura da imagem ou até a ferramenta de mover, para deslocar o desenho em alguma direção.

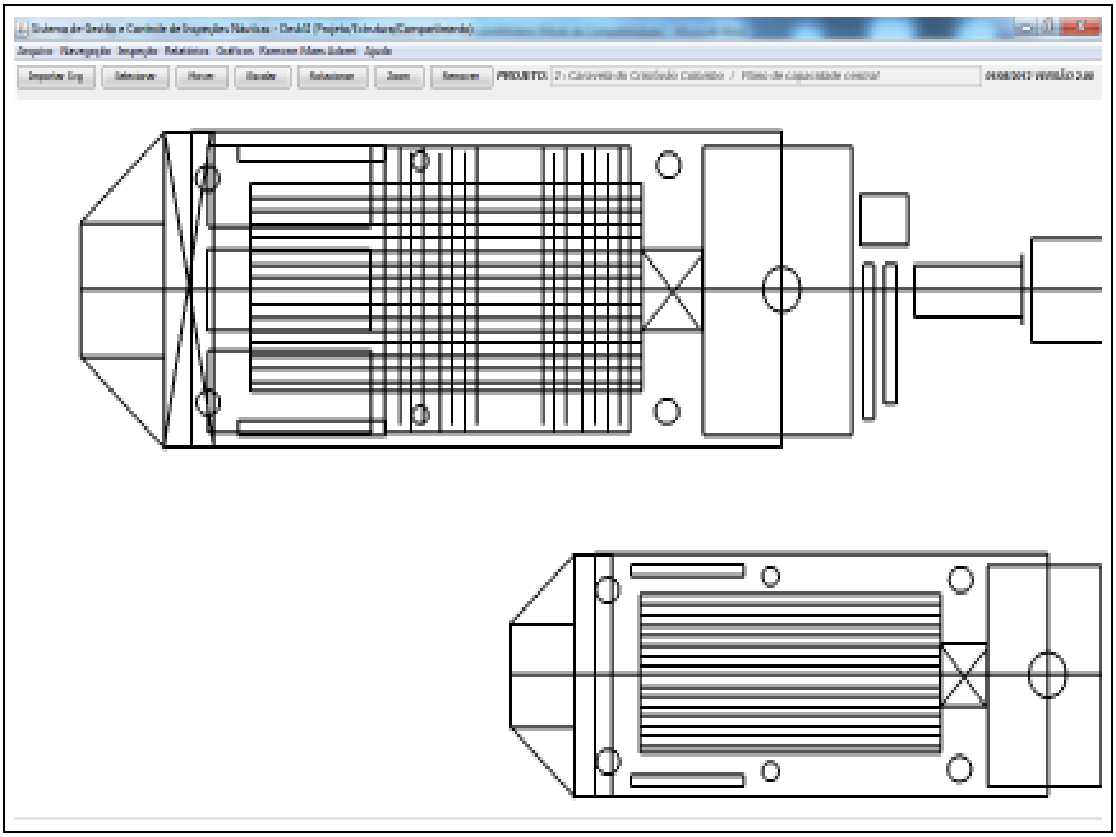


Figura 34 – Tela principal

A Figura 35 mostra a composição dos menus a partir da tela principal, as funções de cada menu são diversificadas e na grande maioria possuem teclas de atalho para facilitar o acesso as suas funções. Estão distribuídos da seguinte forma:

a) `novo projeto`: responsável por abrir a tela de cadastro de projeto;
b) `abrir projeto`: responsável por abrir a tela de seleção de projeto, onde o usuário escolhe o projeto a ser trabalhado no momento;
c) `nova estrutura`: responsável por abrir a tela de cadastro de estrutura;
d) `abrir estrutura`: responsável por abrir a tela de seleção de estrutura onde o usuário escolhe a estrutura a ser trabalhada no momento;
e) `salvar estrutura`: responsável por gravar os dados extraídos da leitura do arquivo, na base de dados. Esta função é aplicada apenas uma vez por estrutura, caso usuário queira incluir um novo desenho, será necessário cadastrar uma nova estrutura e importá-lo;
f) `fechar projeto/estrutura`: responsável por limpar o cenário.
g) `próxima estrutura`: responsável por transitar o frame para a próxima estrutura cadastrada para o projeto aberto;
h) `estrutura anterior`: responsável por transitar o frame para a estrutura anterior

cadastrada para o projeto aberto;

i) `re-montar estrutura`: responsável por renderizar a estrutura da forma que foi extraída do arquivo, serve como desfazer no momento em que o usuário manipula o cenário;

j) `relatório/gráficos`: responsável por abrir a tela de filtros que contém os botões responsáveis pela geração do relatório e dos gráficos;

k) `menu com o nome do usuário`: responsável pelas funções de troca de usuário, sair do sistema e por controlar entrada e saída de novos cadastros de usuários no sistema. A tela que controla o cadastro do usuário tem o mesmo layout e as mesmas funções das outras telas de cadastro;

`ajuda`: mantém informações de versão e pessoa responsável pelo desenvolvimento do sistema.

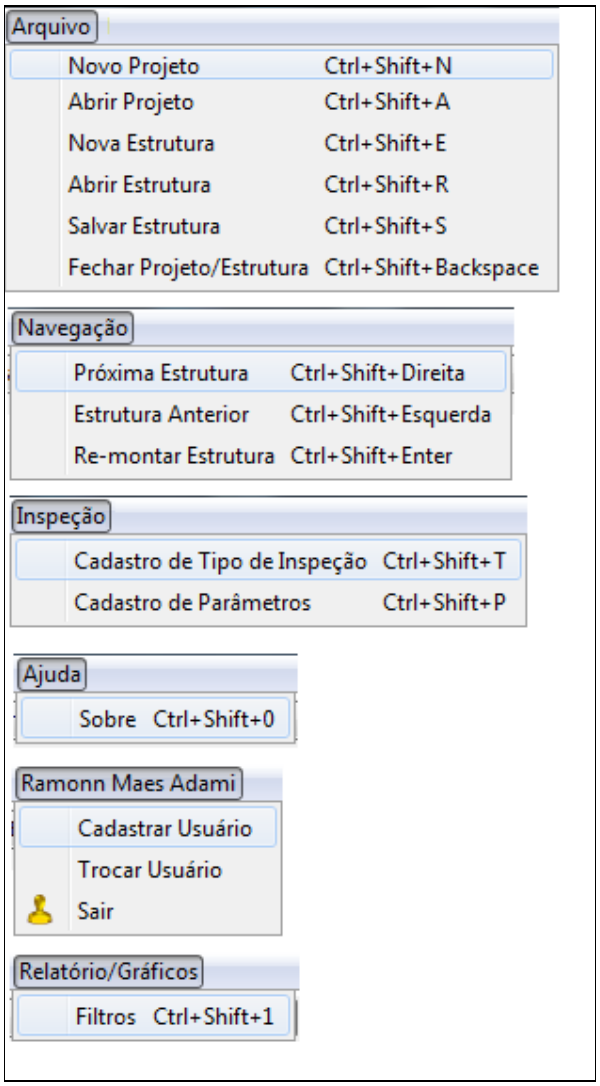


Figura 35 - Menus

Na Figura 36 é exibida a tela de cadastro de projeto, formada pelos campos do cadastro e por um componente de tabela que mantém a exibição dos registros da tabela de projetos. Nesta tela é permitido efetuar uma pesquisa pelo código ou pelo nome do projeto, efetuar uma exclusão ou efetuar uma edição de valores nos campos da tabela.

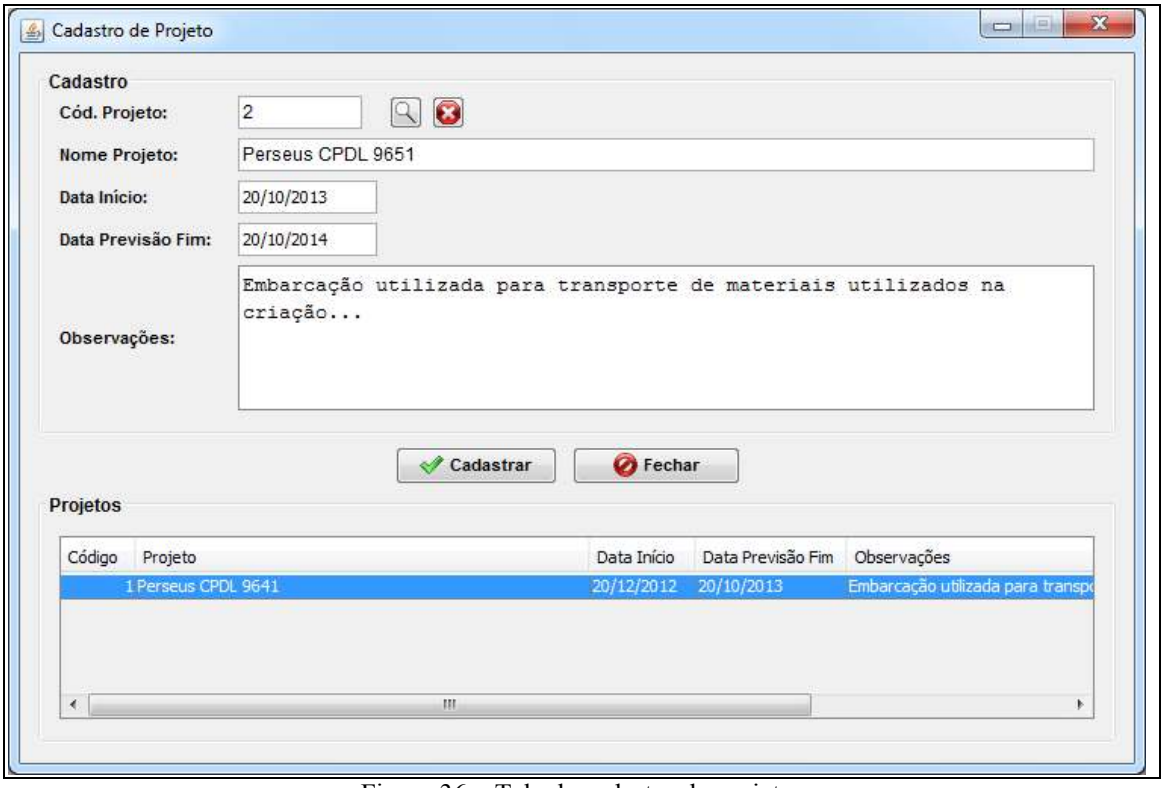


Figura 36 – Tela de cadastro de projeto

Para a abertura ou seleção de um projeto pré-cadastrado, utiliza-se a tela abrir projeto que contém os campos de pesquisa e a tabela de projetos, porém nesta tela é exibido um botão com a função de selecionar o projeto da linha corrente e aplicar como projeto aberto. O usuário tem obrigação de utilizar esta tela para poder continuar o seu fluxo de trabalho no sistema. Esta tela é exibida na Figura 37.

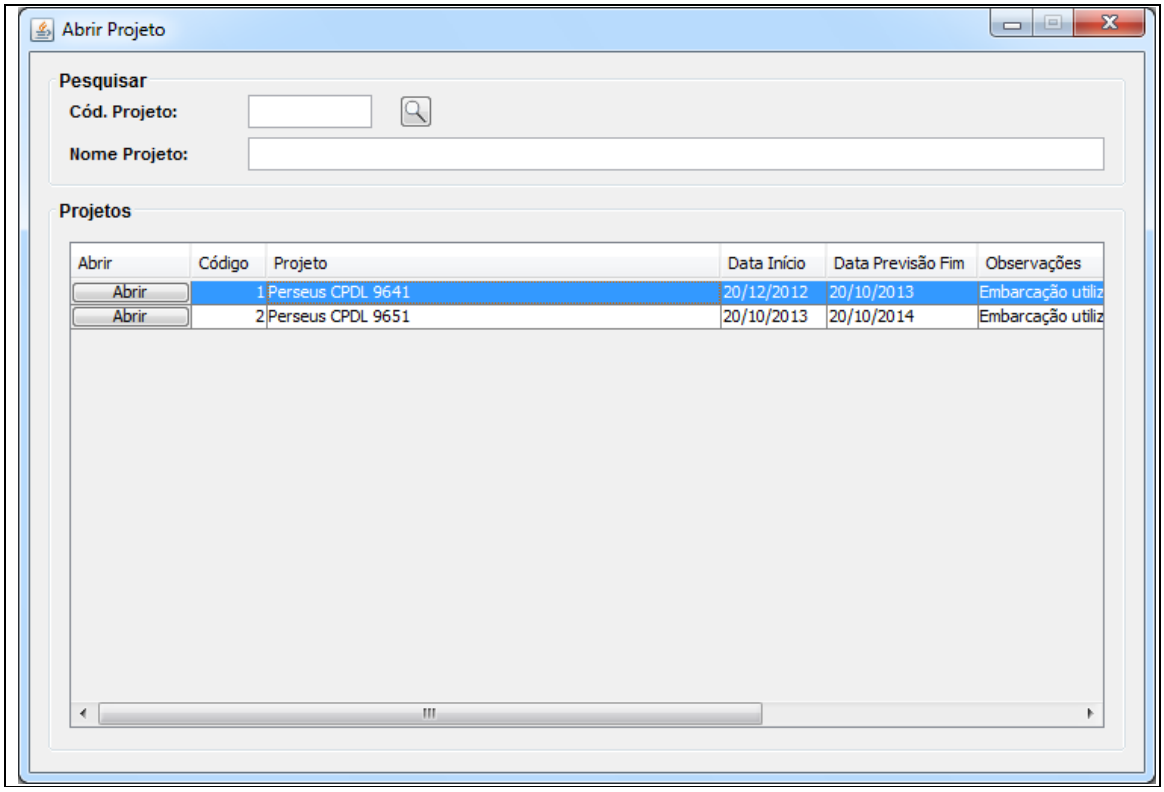


Figura 37 - Tela de seleção de projeto

Após a seleção do projeto é necessário abrir ou cadastrar uma nova estrutura para dar continuidade no processo. A tela responsável pelo cadastro é exibida na Figura 38 e a tela responsável por selecionar a estrutura aberta é mostrada na Figura 39. Estas duas telas seguem um layout padrão com campos de pesquisa e um componente de tabela para facilitar a visualização e edição dos registros cadastrados.

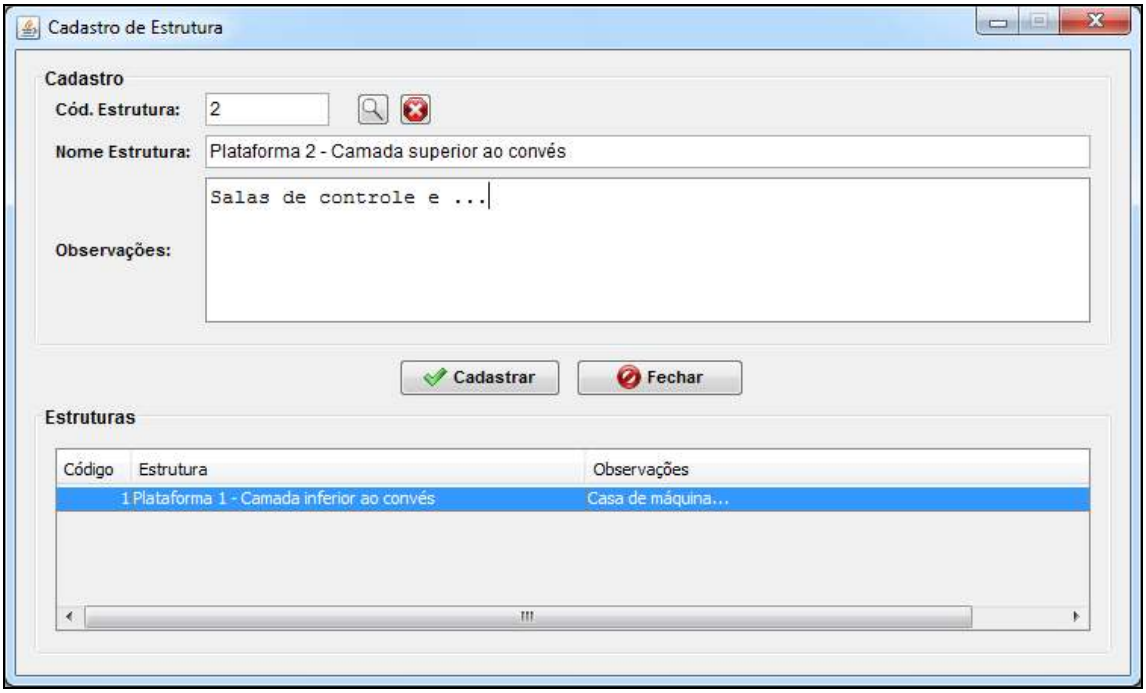


Figura 38 - Tela de cadastro de estrutura

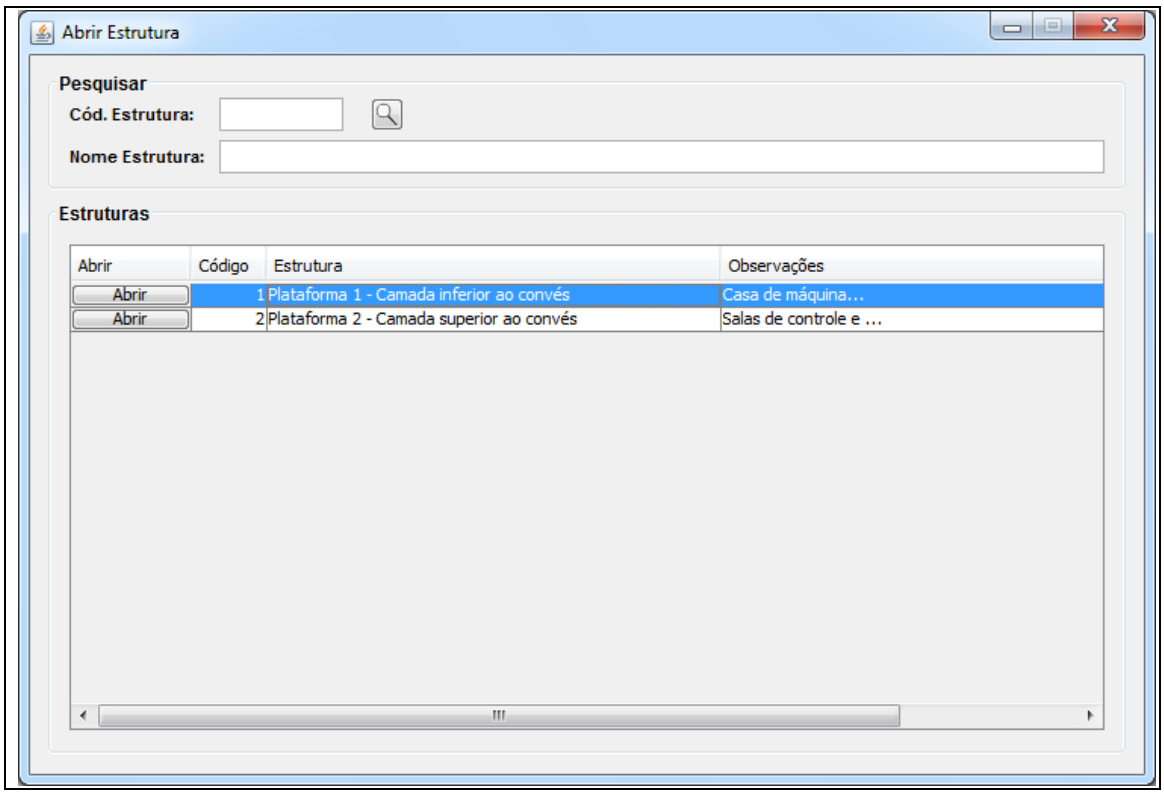


Figura 39 - Tela de seleção de estrutura

O cadastro de compartimento não é acessado via menu e sim logo após seleção do compartimento da estrutura no frame gerado. Esta tela segue os padrões das telas de cadastro de projetos e também de estruturas, porém ela atualiza os dados dos objetos renderizados no

frame principal. Também é responsável pela abertura da tela de registro de inspeções exibida na Figura 40. É neste momento que começa a associação entre o compartimento e a inspeção que será realizada, todas as informações sobre projeto, estrutura e compartimento são passados como parâmetro para a classe de controle responsável por registrar a inspeção na base de dados.

No momento da abertura desta tela, observa-se um comportamento diferente quando um compartimento selecionado já está cadastrado na base de dados ou não, para melhor entendimento segue descrição do funcionamento:

a) usuário efetua um duplo clique no compartimento desejado;
b) sistema abre a tela de cadastro de compartimento;
c) sistema verifica se o compartimento já cadastrado na base;
d) caso passo c retorne falso, a tela habilita os campos do cadastro e o usuário deve informar os dados necessários;
e) após cadastro efetuado a tela desabilita os campos, não permitindo o usuário cadastrar mais de um compartimento por seleção;
f) caso passo c retorne verdadeiro, a tela desabilita os campos do cadastro e seleciona o compartimento já cadastrado na tabela, induzindo o entendimento do usuário a aquele cadastro.

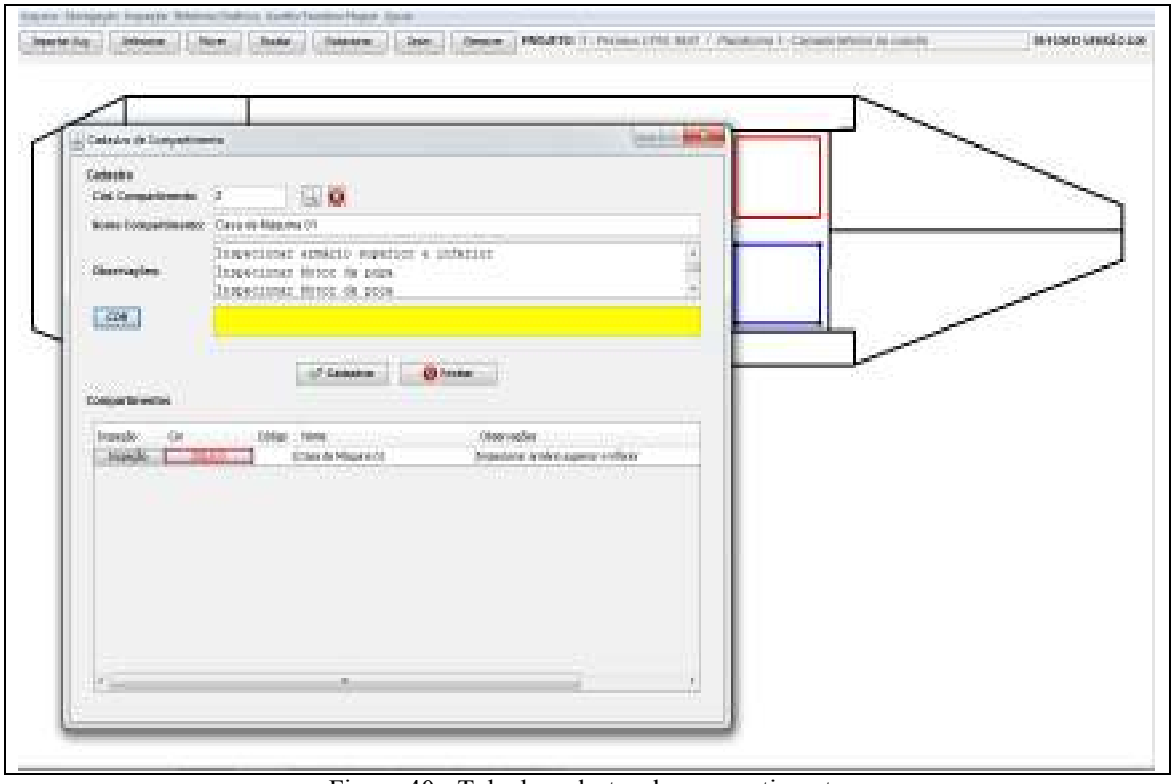


Figura 40 - Tela de cadastro de compartimento

O registro de inspeções (Figura 41) é acessado através do clique no botão “Inspeção” no componente de tabela que armazena os compartimentos cadastrados na tela de cadastro de compartimento. Para a inspeção o usuário informa os dados necessários e clica no botão “Gravar”. Os campos de cadastros são liberados apenas depois do usuário escolher o tipo de inspeção que é obrigatório.

Os botões com rótulo “+” e ”-” servem para incluir uma pendência ou excluir, sendo que quando alterado deve-se gravar as informações novamente.

Os campos do cadastro foram pré-definidos com ajuda do *stakeholder*, sendo que foram levantados como os mais importantes para realização de uma inspeção.

Nota-se também a existência de uma imagem na tela, esta imagem é substituída pelo logotipo da empresa que utiliza o sistema. Os campos de vinculação da inspeção com o projeto/estrutura/compartimento não podem ser alterados devido a integridade dos dados na base.

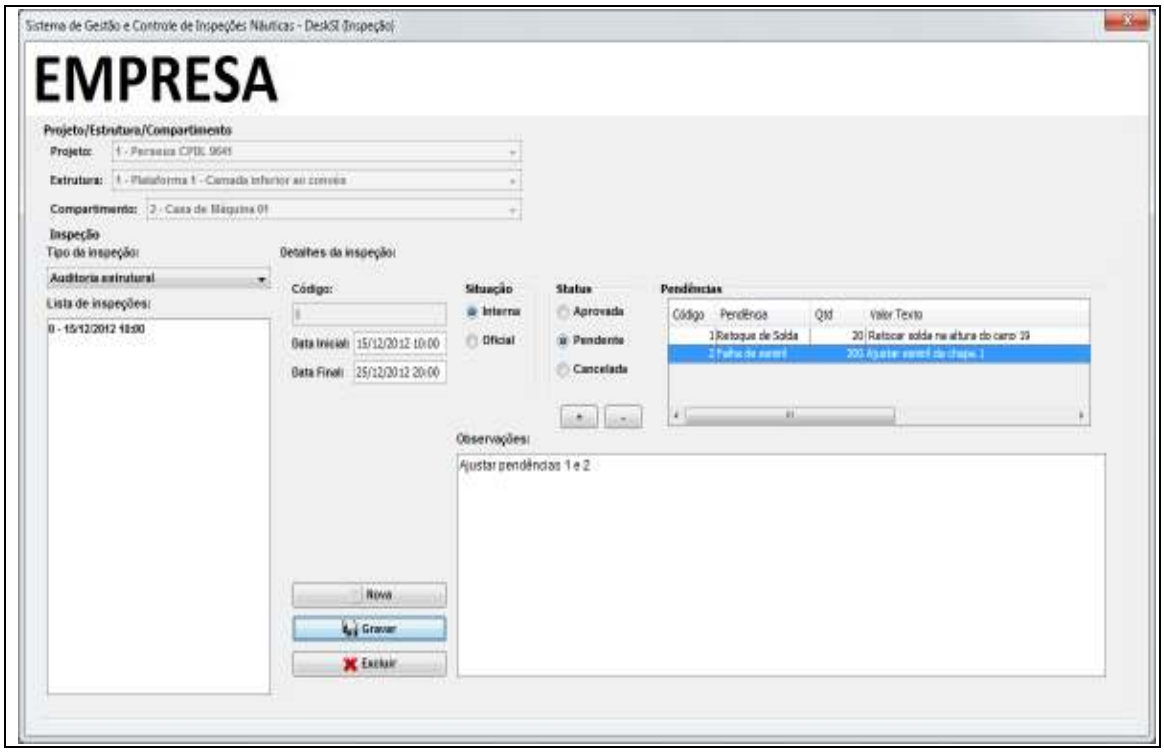


Figura 41 - Tela de registro de inspeções

3.4 RESULTADOS E DISCUSSÃO

Os resultados e discussões estão divididos em três seções. Na seção 3.4.1 é apresentada a avaliação realizada sobre a utilização do sistema, na seção 3.4.2 é descrita a análise dos resultados qualitativos obtidos, na seção 3.4.3 é descrita a análise dos resultados quantitativos obtidos e na seção 3.4.4 é mostrada a comparação entre as funcionalidades do sistema desenvolvido e os trabalhos correlatos.

3.4.1 Avaliação do sistema

Esta seção descreve as avaliações realizadas no sistema desenvolvida, tendo como objetivo principal apresentar a metodologia e os resultados obtidos durante este processo.

Para avaliar o sistema construído foi feito uma demonstração da mesma e aplicado um questionário (Apêndice A) à funcionários que trabalham em determinado estaleiro da cidade de Itajaí do estado de Santa Catarina. Elas efetuam inspeções diariamente e já trabalharam com o protótipo que foi unificado a este trabalho. A avaliação foi aplicada em uma amostra de 3 funcionários sendo que 1 deles tem o cargo de Chefe de Sessão de Controle de Qualidade Nível IV, o outro tem o cargo de Inspetor Técnico Nível IV e o último tem o cargo de Inspetor Técnico Nível I. Os três avaliadores são do sexo masculino com idade entre 21 e 37 anos e estão cursando ou já cursaram o ensino superior. Essa avaliação foi efetuada entre os dias 14 e 19 de novembro do ano de 2012 e para a condução do experimento, optou-se pela utilização do estudo de caráter descritivo, sendo que o questionário aplicado é subdividido em um questionário aberto e outro fechado, que contribuíram para uma análise quantitativa e qualitativa, respectivamente. No questionário aberto deixou-se um espaço onde o usuário pudesse registrar sua opinião sobre o sistema e o questionário fechado correspondeu a uma série de questões específicas sobre a satisfação do usuário em relação à experiência realizada.

O resumo do questionário de avaliação do sistema será exibido nas duas próximas seções onde se observa que a percepção geral dos entrevistados demonstra que o sistema melhoraria o processo de registro de inspeções e facilitaria o desempenho do gestor no gerenciamento de inspeções futuras.

3.4.2 Análise qualitativa dos resultados do questionário aberto

A avaliação do questionário aberto foi feita com base nas anotações do avaliador no momento da entrevista e também com base nas anotações dos entrevistados.

Dentre as anotações feitas pelo avaliador e pelos entrevistados nas sessões de entrevista, destacam-se os itens a seguir:

a) foram feitos comentários positivos e construtivos para a melhoria do sistema de importação da estrutura. Por exemplo, a importação de desenho a partir de arquivos do software AutoCad;
b) foram feitos comentários referente a compartimentos desenhados com polígonos abertos onde é complicado a seleção do mesmo.
c) foi elogiado o desempenho do sistema em relação a velocidade do carregamento dos dados;
d) foram feitos comentários positivos referente as teclas de atalho utilizadas para navegação entre as estruturas;
e) foi identificado um problema na emissão do gráfico de inspeções mensais por funcionário;
f) foi elogiado pelo gestor a estrutura do gráfico de inspeções mensais por projeto;
g) foram feitas sugestões de novos tipos de gráficos, por estrutura e por compartimento;
h) foram feitas sugestões de novos campos de controle de dados nos cadastros e respectivamente no relatório
i) foi feito uma sugestão para integração com software de controle ferramentaria;
j) foram feitas sugestões para criar um botão para enviar o relatório de inspeções por e-mail;
k) foi dado ênfase na possibilidade de controle de pendências em inspeções futuras, onde o funcionário pode se anteceder e modificar a forma de inspeção para não gerar tais pendências.

No geral, os usuários entenderam o fluxo de funcionamento do sistema e a hierarquia de projeto, estrutura, compartimento e inspeção, aplicada. Todas as solicitações do formulário foram atendidas, sendo que algumas com um pouco de dificuldade e outras com muita facilidade.

Dentre as que mais tiveram dificuldade, encontra-se a manipulação da estrutura

importada. Após as instruções os entrevistados conseguiram identificar sem problemas os pontos indicados.

3.4.3 Análise quantitativa dos resultados do questionário fechado

Os usuários também responderam a sete perguntas do questionário fechado. O resultado desse questionário é apresentado no Quadro 4. As opções de respostas observam uma escala de níveis que segue uma gradiente em termos de concordância a discordância. Em cada questão, os usuários assinalaram o grau de avaliação do sistema em termos de "Concordo totalmente", "Concordo parcialmente", "Não concordo nem discordo", "Discordo parcialmente" e "Discordo totalmente". Este tipo de questionário ressalva as respostas positivas, negativas e irrelevantes em relação ao software avaliado. Esse modelo de gradiente foi retirado de Kruger, Fritsch e Viccari (2001, p. 21). As perguntas referem-se a aspectos relativos à usabilidade, funcionalidade e eficiência do sistema avaliado.

A partir das respostas obtidas na avaliação foi formulado o Quadro 4.

Perguntas / Critérios de avaliação	**Concordo totalmente**	**Concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
1. É fácil de encontrar as opções / funcionalidades do sistema	66,66%	33,33%			
2. A estrutura importada para associar as inspeções é de fácil manipulação		66,66%	33,33%		
3. É importante o registro de inspeções	100%				
4. É necessário a integração dos dados com outra ferramenta	33,33%	33,33%	33,33%		
5. É importante a funcionalidade de importar estrutura	33,33%	33,33%	33,33%		
6. É fácil completar o fluxo de trabalho no sistema		100%			
7. De uma maneira geral o sistema é muito bom	100%				

Quadro 4 - Respotas da avaliação do sistema

Após análise dos valores aplicados no Quadro 4, tem-se a idéia de que o sistema foi construído de forma coerente e que induz o usuário à utilização no fluxo correto. Esta afirmação é obtida com a leitura dos valores da questão 1.

A questão 2 demonstra um pouco de dificuldade perante a utilização do frame de controle da estrutura.

Com relação às respostas da questão 3 é possível identificar que os participantes concordam totalmente com a utilização de ferramentas e sistemas para registro de inspeção e,

que elas são de fundamental importância para controle no setor.

Nas questões 4 e 5 nota-se opiniões diferenciadas entre os 3 entrevistados, sendo que a maioria deles concordam em trabalhar com base na estrutura da embarcação porém cada um tem sua opinião referente a importância de manter essa informação em uma base de dados. Em relação a integração com outro sistemas, na seção anterior 3.4.2, foi comentado sobre a integração do sistema desenvolvido com um possível software de controle de ferramentaria.

Com o resultado da questão 6 é possível concluir que os avaliados conseguiram utilizar o sistema para o registro de inspeção, mas que tiveram um pouco de dificuldade, provavelmente em função de ser a primeira vez que estavam utilizando-a.

A partir dos resultados da questão 7 é possível concluir que de maneira geral o sistema teve uma boa aceitação e que não apresentou grandes problemas em sua utilização.

3.4.4 Relação do trabalho atual com os trabalhos correlatos

Com relação aos trabalhos correlatos tem-se o protótipo DeskSI que já estava em funcionamento e influenciou diretamente na facilidade de utilização deste novo sistema pelos funcionários entrevistados. O sistema proposto neste trabalho tem como objetivo ser um sistema mais complexo que o protótipo, que auxilia e facilita a localização e associação da inspeção a ser realizada, através de um frame que reproduz a estrutura física da embarcação. Este frame é o grande diferencial desta nova versão para a versão protótipo DeskSI.

A relação com o sistema ElcoShip fica difícil de se expor devido ao restrito acesso sobre a documentação e especificação do sistema. O sistema ElcoShip não trabalha com a hierarquia Projeto/Estrutura/Compartimento/Inspeção, porém possui uma grande capacidade de registrar os mais variados dados de inspeções conforme padrões das organizações internacionais. Os registros são criados por tarefa, data de área, em análise e por inspetor juntamente com qualquer exigência para o retrabalho ou concessões. Um diferencial importante do sistema ElcoShip que pode ser implementado no sistema desenvolvido neste trabalho, é o sistema de alerta de inspeções. Este sistema auxilia a identificação de inspeções atrasadas e até mesmo de inspeções criadas de forma errada.

Ao analisar o conjunto de características abrangido pelo sistema desenvolvido com relação aos trabalhos correlatos é possível verificar o como pode ser prático de ser utilizado e o quanto ele pode ser importante para esta área específica.

4 CONCLUSÕES

A área naval teve nos últimos anos um grande crescimento e por este motivo, aumentou a demanda de ferramentas especializadas que facilitam a vida das pessoas que trabalham nos estaleiros. Em específico, no setor de controle de qualidade tem-se uma grande deficiência destas ferramentas, principalmente para registros e armazenamento de informações sobre as atividades de inspeções realizadas. O sistema desenvolvido propõe-se a realizar o registro de tais inspeções de forma dinâmica e interativa, através de uma estrutura que contém o desenho técnico ou planta da embarcação.

Em relação aos trabalhos correlatos tem-se o sistema ElcoShip que foi desenvolvido em um padrão para atender os requisitos da organização internacional IMO, onde é possível registrar inspeções realizadas em revestimentos de tanques e desempenhar algumas outras funções de cadastro como fornecedores e atividades diárias.

Em relação ao sistema protótipo DeskSI, o mesmo foi reutilizado e melhorado afim de torná-lo mais interativo. Foi desenvolvido um componente de navegação que usará as informações vetoriais da planta da embarcação representadas por um arquivo de extensão SVG. A interpretação do arquivo SVG foi feita utilizando linguagem JAVA e com base nos marcadores XML. Por fim, foi disponibilizada a tela com o desenho gerado a partir da importação do arquivo. Nesta tela será possível realizar transformações geométricas, cujo objetivo é facilitar a visualização dos compartimentos e associar o registro das inspeções desejadas.

Não foram encontradas dificuldades referente a capacitação dos usuário no momento do teste, devido ao grande período de utilização do protótipo que já está em uso no estaleiro.

Com relação às tecnologias utilizadas, neste trabalho foi utilizado o ambiente de desenvolvimento Eclipse Java *Enterprise Edition*, com a linguagem de programação JAVA e o gerenciador de banco de dados MySQL, sendo que não foram encontrados problemas com a utilização das mesmas, todas colaboraram para a conclusão deste trabalho.

Com o término do desenvolvimento do sistema, conclui-se que os objetivos e os fluxos de trabalho previstos foram alcançados e como resultado, subentende-se uma grande importância na realização dos registros das inspeções através do software desenvolvido neste trabalho.

4.1 LIMITAÇÕES

A principais limitações do sistema são em relação a complexidade dos desenhos utilizados para gerar a estrutura interativa. O formato de arquivo SVG não comporta todos os elementos estruturais que uma planta de uma embarcação gerada através do software AutoCad ou outro similar, pode comportar. A limitação que mais causou impacto no desenvolvimento do trabalho foi o fato de elementos geométricos estruturais serem desenhados utilizando formas geométricas do tipo polígonos aberto, dificultando o desenvolvimento do algoritmo de seleção do compartimento na estrutura.

Outra limitação encontrada é a existência de elementos sobrepondo outros elementos nos desenhos. O algoritmo de seleção foi ajustado para buscar e selecionar o último elemento desenhado, ou seja, não está totalmente de acordo, fazendo com que em algumas situações o usuário tenha que mover os compartimentos com a ferramenta, para então selecionar o correto.

4.2 EXTENSÕES

Como extensão para o trabalho propõe-se:

a) permitir que o sistema faça a importação das coordenadas dos desenhos a partir da leitura de um arquivo com extensão DWG. Este formato pertence à AutoDesk que é fabricante do software AutoCAD, responsável pela geração desse tipo de formato. No atual trabalho, permite apenas a leitura de arquivos com extensão SVG;
b) incluir algoritmos e técnicas para ajustes das coordenadas extraídas através da leitura do arquivo, estes algoritmos devem tratar a possibilidade de seleção de compartimentos criados com elementos que não sejam polígonos convexos, conforme visto nas limitações do trabalho existe uma dificuldade para seleção dos compartimentos criados com elementos de polígonos côncavos;
c) converter o software criado para plataforma Android, possibilitando a utilização do software em PDAs no campo de trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

ARGONAVIS. **Treinamento em tecnologias web e xml**. [S.l.], 2011. Disponível em: <argonavis.com.br>. Acesso em: 06 maio 2012.

BARROS, Leliane N.; SANTOS, Eduardo T. Um estudo sobre a modelagem do domínio de geometria descritiva para a construção de um sistema tutor inteligente. In: XI SIMPÓSIO BRASILEIRO DE INFORMÁTICA EDUCATIVA SBIE, Maceió. **Anais**... Maceió: UFAL, 2000. p. 259-266.

CARVALHO, Marília G. Tecnologia, desenvolvimento social e educação tecnológica. **Revista educação & tecnologia.** Curitiba: Centro Federal de Educação Tecnológica do Paraná, semestral, p. 70-87, jul. 1997.

CHEN, Jianer. **Computational geometry**: methods and applications. Texas: A&M University, 1996. Disponível em: <faculty.cs.tamu.edu/chen/notes/geo.pdf>. Acesso em: 13 jun. 2012.

COHEN, Marcelo; MANSSOUR, Isabel H. **OpenGL**: uma abordagem prática e objetiva. São Paulo: Novatec, 2006.

ELCOSHIP. **ElcoShip**: the solution. [S.l.], 2008. Disponível em: <http://www.elcoship.com/>. Acesso em: 12 maio. 2012.

FARIAS, Bruno V. **Inferência bayesiana de dados de inspeção de estrutura de casco de navio plataforma.** 2010. 136 f. Dissertação (Mestrado em Engenharia Oceânica) – Universidade Federal do Rio de Janeiro / Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia, Rio de Janeiro.

FERNANDES, Cristina G. **Geometria computacional.** [S.l.], 2009. Disponível em: <http://www.ime.usp.br/~cris/aulas/09_2_331/intro.pdf>. Acesso em: 17 jun. 2012.

FERREIRA, Aurélio B. H. **Dicionário novo aurélio da língua portuguesa.** Rio de Janeiro: Nova Fronteira, 1986.

FREITAS, Carla M. D. S. et al. Introdução à visualização de Informações. **Revista de Informática Teórica e Aplicada**, [S.l.], v. 8, n. 2, p. 145, 2001. Disponível em: <http://infovis.ucpel.tche.br/luzzardi/Rita.pdf >. Acesso em: 13 out. 2012.

FONSECA, Maurílio M. **Arte naval / Maurílio Magalhães Fonseca.** 6. ed. Rio de Janeiro: Serviço de Documentação da Marinha, 2002. 873 p.

_____. 5. ed. Rio de Janeiro: Serviço de Documentação da Marinha, 1989. 514 p.

GENTEPRAIAS. **Controle de qualidade total (TQC)**. [S.l.], 2011. Disponível em: <http://www.gentepraias.com.br/colunista.php?id=GNT_mps>. Acesso: em 30 maio 2012.

ITAPORANGASP. **Computação gráfica**. [S.l.], 2011. Disponível em: <http://itaporangasp.com/usp/Computacao_Grafica/slides/SCC0250-slides-13-Preenchimento_poligonos.pdf>. Acesso em: 02 nov. 2012.

KRUGER, Susana E.; FRITSCH, Eloi; VICCARI, Rosa M. Avaliação pedagógica do software STR. **Revista Brasileira de Informática na Educação**, Florianópolis, p. 21-33, nov. 2001.

LEVINE, John. **Programing for graphics files in C and C++**. New York: Wiley, 1993.

MANDARINO, Denis. **Desenho projetivo e geometria descritiva**. São Paulo: Plêiade, 1996.

MCREYNOLDS, Tom; BLYTHE, David. **Advanced graphics programming using OpenGL**. San Francisco: Elsevier, 2005.

MIRANDA, Angela L. **Da natureza da tecnologia:** uma análise filosófica sobre as dimensões ontológica, epistemológica e axiológica da tecnologia moderna. 2002. 161 f. Dissertação (Mestrado em Tecnologia) - Programa de Pós-Graduação em Tecnologia do Centro Federal de Educação Tecnológica do Paraná, Paraná.

MORRISON, Mike. **Mágicas da computação gráfica**. Tradução Elisa M. Ferreira. São Paulo: Berkeley, 1995.

MOUNT, David M. **Computational geometry**. Maryland, 2002. Disponível em: <http://www.cs.umd.edu/~mount/754/Lects/754lects.pdf>. Acesso em: 13 jun. 2012.

MYSQL. **Mysql workbench**. [S.l.]. 2012. Disponível em: <https://www.mysql.com/products/workbench/>. Acesso em: 13 ago. 2012.

OCEÂNICA. **Produto acadêmico:** análise estrutural. [S.l.], 2011. Disponível em: <http://www.oceanica.ufrj.br/deno/prod_academic/relatorios/atuais/DanielW+LeandroTrov/Relat2/05_analise_estrutural.htm>. Acesso em: 12 maio 2012.

OLKUN, Sinan. **Making connections**: improving spatial abilities with engineering drawing activities. International Journal of Mathematics Teaching and Learning. 2003. Disponível em: < http://www.cimt.plymouth.ac.uk/journal/sinanolkun.pdf>. Acesso em: 23 out. 2012.

RAMOS, Evandro de M. Percepção visual e representação gráfica. In: SIMPÓSIO NACIONAL DE GEOMETRIA DESCRITIVA E DESENHO TÉCNICO, 17, 2005. Recife: Universidade Federal de Pernambuco. **Anais**… Recife: Graphica, 2005.

SERUNIVERSITÁRIO. **Engenharia naval e oceânica:** resumo de curso. [S.l.], 2011. Disponível em: <http://www.seruniversitario.com.br/cursos-graduacao/cursoSuperior.php?ac=engenharia-naval-e-oceanica&id=74>. Acesso em: 13 jun. 2012.

SHREINER, Dave et al. **OpenGL programming guide**. 5th ed. [S.l.]: Addison-Wesley, 2005. Disponível em: <http://www.gamedev.net/download/redbook.pdf>. Acesso em: 02 abr. 2012.

SILVA, Marco A. N. **Mineração visual de dados:** extração do conhecimento a partir das técnicas de visualização da informação e mineração de dados. 2008. 171 f. Dissertação (Mestrado em Ciências na área de concentração Programação Matemática) - Programa de Pós-Graduação em Métodos Numéricos em Engenharia (PPGMNE), Universidade Federal do Paraná, Paraná. Disponível em: <http://www.ppgmne.ufpr.br/arquivos/diss/191.pdf>. Acesso em: 03 nov. 2011.

TAN, Pang-Ning; STEINBACH, Michael; KUMAR, Vipin. **Introduction to data mining**. Boston: Addison-Wesley, 2006.

VIEIRA, Jéssica M. L.; CORRÊA, Renato F. Recuperação de informação através de recursos visuais. In: ENCONTRO NACIONAL DE ESTUDANTES DE BIBLIOTECONOMIA, DOCUMENTAÇÃO, GESTÃO, E CIÊNCIA DA INFORMAÇÃO, 33., 2010, João Pessoa. **Anais...** João Pessoa: Universidade Federal da Paraíba. Não paginado. Disponível em: <http://dci.ccsa.ufpb.br/enebd/index.php/enebd/article/viewFile/19/22>. Acesso em: 20 out. 2011.

W3C. **XML Tutorials**: svg. [S.l.], 2012. Disponível em: <http://www.w3schools.com/>. Acesso em: 14 set. 2012.

APÊNDICE A – Questionário de avaliação

No Quadro 5 é exibido o questionário de avaliação do sistema.

Data de realização da avaliação: / /2012.

Observação: As informações recebidas serão mantidas confidenciais.

PRIMEIRA PARTE:
Este questionário possui 6 questões. Por gentileza, responda cada uma delas.
Preencha o seguinte questionário de informações sobre o seu perfil. Seja o mais específico possível nas respostas.

1. Sexo: ( ) Masculino ( ) Feminino
2. Idade: ___ anos.
3. Profissão atual:
4. Tempo na profissão: ___ anos.
5. Já utilizou/utiliza algum sistema para o registro dos dados das inpeções:
   ( ) Sim ( )Não
6. Quais sistemas já utilizou (Opcional):

SEGUNDA PARTE:
O objetivo desta parte do experimento é identificar se os usuários têm facilidade para utilizar o sistema de registro de inspeções desenvolvido. Para isso foi preparado um roteiro organizado por etapas, instruindo o usuário a realizar procedimentos que exploram as funcionalidades do sistema. O roteiro foi separado em duas etapas. Na primeira etapa foi solicitada desde a criação de um novo projeto até a importação da estrutura do arquivo. Esta etapa está voltada mais para utilização pelos gestores.

Etapa 1: Realize os procedimentos conforme descritos a seguir:
1. acesse o sistema, será apresentada a tela de login do sistema;
2. selecione seu usuário e informe a sua senha;
3. identifique onde são criados os novos projetos e faça o cadastro de um novo;
4. para criar o projeto informe um código numérico maior que zero, dê um nome de sua escolha, informe a data de início e previsão de fim, deixe a observação em branco.
5. identifique o botão “Cadastrar” e efetue o cadastro;
6. o projeto será criado e o componente de tabela logo abaixo dos campos de cadastro será atualizado;
7. identifique o botão para fechar a tela;
8. identifique no menu onde são selecionados/abertos os projetos para o trabalho e faça a seleção do projeto que foi cadastrado, descubra como fazer a seleção do projeto;
9. o campo projeto na tela principal será atualizado e a tela de seleção será fechada;
10. identifique onde são cadastradas as estruturas e faça o cadastro de uma nova;
11. para criar a estrutura, informe um código numérico maior que zero e o nome da estrutura, o campo observação deixe em branco;
12. identifique o botão “Cadastrar” e efetue o cadastro;
13. a estrutura será criada e o componente de tabela logo abaixo dos campos de cadastro será atualizado;

14. identifique no menu onde são selecionadas/abertas as estruturas e faça a seleção desta que foi cadastrada. Descubra como fazer a seleção. Caso já exista um desenho definido para a estrutura, este será carregado na tela principal;
15. o campo Projeto na tela principal será atualizado e a tela de seleção será fechada;
16. após estes passos o botão de importação do arquivo permitirá a seleção do arquivo SVG contendo a estrutura;
17. identifique o botão de importação e selecione o arquivo Camada1.svg;
18. aguarde o término da importação;
19. identifique o menu Salvar estrutura e efetue a gravação.

Qual é a sua opinião sobre o sistema quanto ao seu uso e funcionalidades?
(críticas e sugestões)

_____
_____
_____

Período de realização da etapa 1:
Data de início: / / 2012 Data de fim: / / 2012
Hora de inicio: Hora de fim:

A segunda etapa tem como pré-requisito a realização da primeira etapa.
Na segunda etapa foi solicitada desde a abertura do projeto até o registro e vinculação da inspeção realizada com sucesso. Esta etapa pode ser realizada por todos os atores envolvidos.

Etapa 2: Realize os procedimentos conforme descritos a seguir:
1. identifique o compartimento desejado;
2. descubra como abrir a tela de cadastro de compartimento;
3. efetue o cadastro do compartimento. Caso ele já esteja cadastrado irá trazer a linha no componente de tabela selecionado com o respectivo compartimento;
4. para criar o cadastro do compartimento informe um código numérico maior que zero, informe o nome do compartimento e a cor. O campo observação deixe em branco;
5. o compartimento será cadastrado, o componente de tabela será atualizado e a cor do compartimento será atualizada no desenho;
6. descubra como efetuar uma inspeção para o compartimento cadastrado;
7. para criar uma inspeção será necessário informar os dados coletados pelo inspetor;
8. identifique o botão Gravar e efetue a gravação dos dados;
9. descubra como inserir um pendência para a inspeção criada;
10. inclua uma pendência e aplique o passo 8.

Qual é a sua opinião sobre o sistema quanto ao seu uso e funcionalidades?
(críticas e sugestões)

_____
_____
_____

Período de realização da etapa 2:
Data de início: / / 2012 Data de fim: / / 2012
Hora de inicio: Hora de fim:

TERCEIRA PARTE:
Após a utilização do sistema realizado na segunda parte, você está convidado a responder um questionário de avaliação do sistema.
As respostas deverão ser feitas na tabela abaixo observando às impressões obtidas com a utilização do sistema. Você deve responder preenchendo uma das alternativas.
Após o questionário com perguntas objetivas é apresentado um espaço para comentários gerais sobre o sistema e para sugestões de melhorias.

Perguntas / Critérios de avaliação	**Concordo totalmente**	**Concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
1. É fácil encontrar as opções / funcionalidades do sistema					
2. A estrutura importada para associar as inspeções é de fácil manipulação					
3. É importante o registro de inspeções					
4. É necessário a integração dos dados com outra ferramenta					
5. É importante a funcionalidade de importar estrutura					
6. É fácil completar o fluxo de trabalho no sistema					
7. De uma maneira geral o sistema é muito bom					

Qual é a sua opinião sobre o sistema quanto ao seu uso e funcionalidades?
(críticas e sugestões)
_____
_____
_____

Obrigado pela sua participação!

Quadro 5 - Questionário de avaliação do sistema

APÊNDICE B – Detalhamento de casos de uso

No Quadro 6 é demonstrado o detalhamento do cadastro de usuário.

UC01 - Cadastrar usuário.	
Cenário principal	01) O gestor clica no menu cadastrar usuário 02) O sistema abre a tela de cadastro 03) O gestor informa os dados do usuário 04) O sistema verifica se as informações estão válidas 05) O sistema insere o novo usuário na tabela do banco de dados 06) O sistema atualiza a componente tabela na janela
Cenário alternativo 1	No passo 03, o gestor escolhe excluir um registro: 03.1)O gestor seleciona o registro no componente tabela 03.2)O gestor clica no botão excluir. 03.3)O sistema exclui o registro da tabela do banco de dados 03.4)O sistema atualiza o componente tabela
Cenário alternativo 2	No passo 03, o gestor escolhe editar um registro: 03.1)O gestor seleciona a linha do componente tabela 03.2)O gestor informa os novos dados dentro dos campos do componente 03.3)O sistema verifica se as informações estão válidas 03.4)O sistema atualiza o usuário na tabela do banco de dados 03.5)O sistema atualiza o componente de tabela na janela
Cenário alternativo 3	No passo 03, o gestor escolhe pesquisar um registro: 03.1)O gestor informa o código ou nome do usuário 03.2)O gestor clica no botão pesquisar 03.3)O sistema localiza o usuário e seleciona a linha da tabela correspondente ao registro pesquisado
Cenário exceção 1	Nos passos 04 e 03.3 caso retorne uma exceção: O sistema emite uma mensagem para o usuário com o texto da exceção tratada
Pós-condição	O usuário estará cadastrado, excluído ou selecionado

Quadro 6 - Caso de uso 01 - cadastrar usuário

No Quadro 7 é demonstrado o detalhamento do cadastro de projeto.

UC02 - Cadastrar projeto.	
Cenário principal	01) O gestor clica no menu cadastrar projeto 02) O sistema abre a tela de cadastro 03) O gestor informa os dados do projeto 04) O sistema verifica se as informações estão válidas 05) O sistema insere o novo projeto na tabela do banco de dados 06) O sistema atualiza o componente de tabela na janela
Cenário alternativo 1	No passo 03, o gestor escolhe excluir um registro: 03.1)O gestor seleciona o registro na lista de projetos 03.2)O gestor clica no botão excluir. 03.3)O sistema exclui o registro da tabela do banco de dados 03.4)O sistema atualiza o componente tabela
Cenário alternativo 2	No passo 03, o gestor escolhe editar um registro: 03.1)O gestor seleciona a linha da lista de projetos 03.2)O gestor informa os novos dados dentro dos campos da lista 03.3)O sistema verifica se as informações estão válidas 03.4)O sistema atualiza o projeto na tabela do banco de dados 03.5)O sistema atualiza o componente de lista na janela
Cenário alternativo 3	No passo 03, o gestor escolhe pesquisar um registro: 03.1)O gestor informa o código ou nome do projeto 03.2)O gestor clica no botão pesquisar 03.3)O sistema localiza o projeto e seleciona a linha de projeto correspondente ao registro pesquisado
Cenário exceção 1	Nos passos 04 e 03.3 caso retorne uma exceção: O sistema emite uma mensagem para o usuário com o texto da exceção tratada
Pós-condição	O projeto estará cadastrado, excluído ou selecionado

Quadro 7 - Caso de uso 02 - cadastrar projeto

No Quadro 8 é demonstrado o detalhamento do cadastro de tipo de inspeção.

UC03 - Cadastrar tipo de inspeção.	
Cenário principal	01) O gestor clica no menu cadastrar tipo de inspeção 02) O sistema abre a tela de cadastro 03) O gestor informa os dados do tipo de inspeção 04) O sistema verifica se as informações estão válidas 05) O sistema insere o novo tipo de inspeção na tabela do banco de dados 06) O sistema atualiza o componente de tabela na janela
Cenário alternativo 1	No passo 03, o gestor escolhe excluir um registro: 03.1)O gestor seleciona o registro na lista de tipo de inspeção 03.2)O gestor clica no botão excluir. 03.3)O sistema exclui o registro da tabela do banco de dados 03.4)O sistema atualiza o componente lista
Cenário alternativo 2	No passo 03, o gestor escolhe editar um registro: 03.1)O gestor seleciona a linha da lista de tipo de inspeção 03.2)O gestor informa os novos dados dentro dos campos do componente 03.3)O sistema verifica se as informações estão válidas 03.4)O sistema atualiza o tipo de inspeção na tabela do banco de dados 03.5)O sistema atualiza o componente de lista na janela
Cenário alternativo 3	No passo 03, o gestor escolhe pesquisar um registro: 03.1)O gestor informa o código ou nome do tipo de inspeção 03.2)O gestor clica no botão pesquisar 03.3)O sistema localiza o tipo de inspeção e seleciona a linha da lista correspondente ao registro pesquisado
Cenário exceção 1	Nos passos 04 e 03.3 caso retorne uma exceção: O sistema emite uma mensagem para o usuário com o texto da exceção tratada
Pós-condição	O tipo de inspeção estará cadastrado, excluído ou selecionado

Quadro 8 - Caso de uso 03 - cadastrar tipo de inspeção

No Quadro 9 é demonstrado o detalhamento do cadastro de estrutura. Quando é mencionada a palavra usuário, considerar os atores `gestor` e `funcionário`.

UC04 - Cadastrar estrutura.	
Pré-condição	01) UC02 – Cadastrar projeto
Cenário principal	01) O usuário abre um projeto 02) O usuário clica no menu cadastrar estrutura 03) O sistema abre a tela de cadastro 04) O usuário informa os dados da estrutura 05) O sistema verifica se as informações estão válidas 06) O sistema insere a nova estrutura na tabela do banco de dados 07) O sistema atualiza o componente de lista na janela
Cenário alternativo 1	No passo 04, o usuário escolhe excluir um registro: 04.1)O usuário seleciona o registro na lista de estruturas 04.2)O usuário clica no botão excluir. 04.3)O sistema exclui o registro da tabela do banco de dados 04.4)O sistema atualiza o componente lista
Cenário alternativo 2	No passo 04, o usuário escolhe editar um registro: 04.1)O usuário seleciona a linha da lista de estruturas 04.2)O usuário informa os novos dados dentro dos campos do componente 04.3)O sistema verifica se as informações estão válidas 04.4)O sistema atualiza a estrutura na tabela do banco de dados 04.5)O sistema atualiza o componente de lista na janela
Cenário alternativo 3	No passo 04, o usuário escolhe pesquisar um registro: 04.1)O usuário informa o código ou nome da estrutura 04.2)O usuário clica no botão pesquisar 04.3)O sistema localiza a estrutura e seleciona a linha da lista correspondente ao registro pesquisado
Cenário exceção 1	Nos passos 05 e 04.3 caso retorne uma exceção: O sistema emite uma mensagem para o usuário com o texto da exceção tratada
Pós-condição	A estrutura estará cadastrada, excluída ou selecionada na lista

Quadro 9 - Caso de uso 04 - cadastrar estrutura

No Quadro 10 é demonstrado o detalhamento do cadastro de pendência.

UC08 - Cadastrar pendência.	
Pré-condição	01) UC07 – Registrar inspeção
Cenário principal	01) O usuário clica no botão com o rótulo "+" 02) O sistema reserva um registro de pendência 03) O sistema atualiza o componente de lista na janela de registro de inspeção 04) O usuário atualiza os campos da tabela com as informações necessárias para a pendência 05) O sistema verifica se as informações estão válidas 06) O sistema insere a pendência na tabela do banco de dados 07) O sistema atualiza o componente de lista
Cenário alternativo 1	No passo 01, o usuário clica no botão com rótulo "-" 01.1)O sistema remove o registro selecionado na lista de pendências 01.2)O sistema remove o registro da tabela do banco de dados componente de lista da janela. 01.3)O sistema atualiza o componente de lista na janela
Cenário exceção 1	Nos passos 05 caso retorne uma exceção: O sistema emite uma mensagem para o usuário com o texto da exceção tratada
Pós-condição	O sistema mostra a informação de pendências cadastradas

Quadro 10 - Caso de uso 08 - cadastrar pendência

No Quadro 11 é demonstrado o detalhamento da função de visualizar inspeção.

UC09 - Visualizar inspeção.	
Pré-condição	01) UC07 – Registrar inspeção
Cenário principal	01) O usuário abre um projeto 02) O usuário abre uma estrutura 03) O usuário seleciona um compartimento cadastrado 04) O usuário clica no botão Inspeção na tela cadastro de compartimento 05) O sistema abre a tela de registro de inspeção 06) O usuário seleciona o tipo de inspeção 07) O sistema carrega a lista de inspeções já registradas 08) O sistema habilita os campos de cadastro 09) O usuário clica na inspeção na lista de inspeções 10) O sistema carrega os dados da inspeção seleciona no respectivos campos
Pós-condição	O sistema mostra as informações da inspeção selecionada, nos respectivos campos

Quadro 11 - Caso de uso 09 - visualizar inspeção

No Quadro 12 é demonstrado o detalhamento da função de geração de gráficos.

UC10 - Gerar gráfico/detalhe.	
Cenário principal	01) O usuário clica no menu gráfico 02) O sistema abre a tela contendo as listas com os filtros definidos 03) O usuário seleciona os filtros desejados 04) O sistema executa consultas na base de dados 05) O sistema passa as consultas como parâmetro para a biblioteca JFreeChart 06) A biblioteca abre a janela do gráfico
Pós-condição	O sistema mostra o gráfico em uma nova janela.

Quadro 12 - Caso de uso 10 - visualizar gráfico

No Quadro 13 é demonstrado o detalhamento da função de geração de relatório.

UC11 - Gerar relatório de informações das inspeções	
Cenário principal	07) O usuário clica no menu relatório 08) O sistema abre a tela contendo as listas com os filtros definidos 09) O usuário seleciona os filtros desejados 10) O sistema executa consultas na base de dados 11) O sistema monta um texto contendo as informações das inspeções retornadas na consulta 12) O sistema gera um arquivo texto com as informações coletadas
Pós-condição	O sistema mostra o relatório gerado

Quadro 13 - Caso de uso 11 - visualizar relatório

APÊNDICE C – Dicionário de dados

O dicionário de dados é exibido entre os quadros 14 e 25, ele descreve o nome das tabelas, os campos e uma breve descrição. A seguir o detalhamento.

Tabela: USUARIO			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_USUARIO	Int(11)	Codigo do usuario	Sim
NOME_LOGIN	Varchar(100)	Nome de login	
NOME_USUARIO	Varchar(100)	Nome de usuario	
SENHA_ACESSO	Varchar(20)	Senha de acesso	
GESTOR	Varchar(5)	É gestor ?	
DESC_OBSERVACAO	Varchar(4000)	Descricao da obsevacao	

Quadro 144 – Dicionário de dados – tabela `usuario`

Tabela: PENDENCIA_INSPECAO			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_PENDENCIA	Int(11)	Codigo da pendencia	Sim
DESC_PENDENCIA	Varchar(4000)	Descricao da pendencia	
QTD_PENDENCIA	Double	Quantidade da pendencia	
VALOR_PENDENCIA_TEXTO	Varchar(100)	Valor da pendencia	
VALOR_PENDENCIA_NUMERO	Double	Valor da pendencia	
ID_INSPECAO	Int(11)	Codigo da inspecao	

Quadro 155 – Dicionário de dados – tabela `pendencia_inspecao`

Tabela: PARAMETRO_INSPECAO			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_PARAMETRO	Int(11)	Codigo do parametro	S
ID_INSPECAO	Int(11)	Codigo da inspecao	
NOM_PARAMETRO	Varchar(100)	Nome do parametro	
TIPO_VALOR	Varchar(20)	Tipo do valor	
VAL_PARAMETRO	Varchar(50)	Valor do paramentro	

Quadro 166 – Dicionário de dados – tabela `parametro_inspecao`

Tabela: INSPECAO			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_INSPECAO	Int(11)	Codigo da inspecao	S
ID_PROJETO	Int(11)	Codigo do projeto	S
ID_ESTRUTURA	Int(11)	Codigo da estrutura	S
ID_COMPARTIMENTO	Int(11)	Codigo do compartimento	S
ID_TIPO_INSPECAO	Int(11)	Codigo do tipo de inspecao	

DATA_INICIAL	Varchar(10)	Data inicial	
HORA_INICIAL	Varchar(8)	Hora inicial	
DATA_FINAL	Varchar(10)	Data final	
HORA_FINAL	Varchar(8)	Hora final	
SITUACAO	Int(11)	Codigo da situacao	
STATUS	Int(11)	Codigo do status	
DESC_OBSERVACAO	Varchar(4000)	Descricao da observação	
ID_USUARIO	Int(11)	Codigo do usuario	

Quadro 177 – Dicionário de dados – tabela `inspecao`

Tabela: TIPO_INSPECAO			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_TIPO_INSPECAO	Int(11)	Codigo do tipo de inspecao	S
NOM_TIPO_INSPECAO	Varchar(50)	Nome do tipo de inspecao	
DESC_SCRIPT	Varchar(4000)	Descricao do script de aplicacao	

Quadro 188 – Dicionário de dados – tabela `tipo_inspecao`

Tabela: ELEMENTO_PONTO_SVG			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_ELEMENTO_PONTO	Int(11)	Codigo do elemento ponto	S
ID_ELEMENTO_ESTRUTURA	Int(11)	Codigo do elemento estrutura	
X	Float	Valor auxiliar	
Y	Float	Valor auxiliar	

Quadro 199 – Dicionário de dados – tabela `elemento_ponto_svg`

Tabela: ELEMENTO_ESTRUTURA_SVG			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_ELEMENTO_ESTRUTURA	Int(11)	Codigo do elemento	S
ID_ESTRUTURA_PROJETO	Int(11)	Codigo da estrutura	S
NR_ORDEM_DESENHO	Int(11)	Numero de ordem do desenho	
DS_CLASSE	Varchar(50)	Descricao da classe do elemento	
NR_CAMADA_DESENHO	Int(11)	Numero de ordem do desenho	
X	Float	Valor auxiliar	
Y	Float	Valor auxiliar	
WIDTH	Float	Valor auxiliar	
HEIGHT	Float	Valor auxiliar	
RX	Float	Valor auxiliar	
RY	Float	Valor auxiliar	
X1	Float	Valor auxiliar	
X2	Float	Valor auxiliar	

Y1	Float	Valor auxiliar	
Y2	Float	Valor auxiliar	
CX	Float	Valor auxiliar	
CY	Float	Valor auxiliar	
R	Float	Valor auxiliar	
ID_PROJETO	Int(11)	Codigo do projeto	
ID_COMPARTIMENTO	Int(11)	Codigo do compartimento	

Quadro 20 – Dicionário de dados – tabela `elemento_estrutura_svg`

Tabela: COMPARTIMENTO_ESTRUTURA_PROJETO			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_COMPARTIMENTO	Int(11)	Codigo do compartimento	S
ID_ESTRUTURA_PROJETO	Int(11)	Codigo da estrutura	S
ID_PROJETO	Int(11)	Codigo do projeto	S
NOM_COMPARTIMENTO	Varchar(60)	Nome do compartimento	
DESC_OBSERVACAO	Varchar(4000)	Descricao da observação	
COR_COMPARTIMENTO	Varchar(45)	Cor do compartimento	

Quadro 21 – Dicionário de dados – tabela `compartimento_estrutura_projeto`

Tabela: ESTRUTURA_PROJETO			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_ESTRUTURA_PROJETO	Int(11)	Codigo da estrutura	S
NOM_ESTRUTURA_PROJETO	Varchar(60)	Nome da estrutura	
ID_PROJETO	Int(11)	Codigo do projeto	
DESC _OBSERVACAO	Varchar(4000)	Descricao da observação	

Quadro 22 – Dicionário de dados – tabela `estrutura_projeto`

Tabela: PROJETO			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_PROJETO	Int(11)	Codigo do projeto	S
DESC_PROJETO	Varchar(100)	Descricao do nome do projeto	
DATA_INICIO	Varchar(10)	Data de inicio do projeto	
DATA_PREVISAO_FIM	Varchar(10)	Data de previsao de fim do projeto	
DESC_OBSERVACAO	Varchar(4000)	Descricao_da observação	

Quadro 23 – Dicionário de dados – tabela `projeto`

Tabela: SITUACAO_INSPECAO			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_SITUACAO	Int(11)	Codigo da situacao da inspecao	S
DESC_SITUACAO	Varchar(50)	Descricao da situacao da inspecao	

Quadro 24 – Dicionário de dados – tabela `situacao_inspecao`

Tabela: tipo_inspecao			
Nome da Coluna	**Tipo de Dados**	**Comentários**	**Chave**
ID_TIPO	Int(11)	Codigo do tipo de inspecao	S
DESC_TIPO	Varchar(50)	Descricao do tipo de inspeção	

Quadro 25 – Dicionário de dados – tabela `tipo_inspecao`