UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

FERRAMENTA PARA ANÁLISE DE LOGS DO TESTCOMPLETE

ALBINO MARCOS DE ANDRADE JUNIOR

BLUMENAU
2015

2015/1-01

ALBINO MARCOS DE ANDRADE JUNIOR

FERRAMENTA PARA ANÁLISE DE LOGS DO TESTCOMPLETE

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 Sistemas de Informação — Bacharelado.

Prof. Jacques Robert Heckmann, Mestre – Orientador

BLUMENAU
2015

2015/1-01

FERRAMENTA PARA ANÁLISE DE LOGS DO TESTCOMPLETE

Por

ALBINO MARCOS DE ANDRADE JUNIOR

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

Presidente:	Prof. Jacques Robert Heckmann, Mestre – Orientador, FURB
Membro:	Prof. Everaldo Artur Grahl, Mestre – FURB
Membro:	Prof. Wilson Pedro Carli, Mestre – FURB

Blumenau, 06 de julho de 2015.

Dedico este trabalho a minha família, aos amigos, e especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, que sempre esteve presente.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, professor Jacques Robert Heckmann, por ter acreditado na conclusão deste trabalho.

Aos professores do Departamento de Sistemas e Computação da Universidade Regional de Blumenau por suas contribuições durante os semestres letivos.

Seja um estudante, não um seguidor. Não vá simplesmente fazer o que alguém diz. Tenha interesse pelo que alguém diz, então debata, pondere e considere de todos os ângulos.

Jim Rohn

RESUMO

Este trabalho apresenta o desenvolvimento de uma ferramenta para analisar os *Test Logs* das execuções de testes automatizados da ferramenta *TestComplete*. Embora o *TestComplete* disponibilize a visualização do *Test Log*, através do qual é possível visualizar detalhadamente passo a passo o que aconteceu em cada execução, ele não permite a visualização de mais de um *Test Log* por vez, dificultando a análise e comparação de determinadas execuções. A ferramenta permite ao usuário importar os dados dos arquivos de *Test Logs* e visualizar os projetos dos seus respectivos responsáveis, além de visualizar as informações dos *Test Logs* importados, com a possibilidade de gerar gráficos com informações referentes aos projetos e seus respectivos casos de testes, e gráficos com informações das execuções dos scripts de testes para auxiliar a visualização. Para o desenvolvimento foi utilizado o Netbeans *Integrated Development Environment* (IDE) e o *framework JFreeChart* para geração dos gráficos.

Palavras-chave: *TestComplete*. *Test Logs*. Testes Automatizados.

ABSTRACT

This work presents the development of a tool to analyze the Test Logs of TestComplete automated tests' run tool. Although TestComplete offers visualization of its Test Log, through which it is possible to visualize in detail step by step what happened in each run, it does not allow the visualization of more than a Test Log a time, making it difficult to analyze and compare specific test runs. The prototype allows the user to extract the data from the files of the Test Logs and to visualize the projects of its respective responsibles, besides visualizing the imported Test Logs information, with the generating graphics possibility with information related to the projetcts and their test cases, and graphics whit information of test runs scripts to assist the visualization. To develop it, Netbeans Integrated Development Environment (IDE) and JFreeChart framework was used.

Keywords: *TestComplete*. *Test Logs*. Automated Testing.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

GUI – *Graphical User Interface*

IDE – *Integrated Development Environment*

MER – Modelo Entidade Relacionamento

SQL – *Structured Query Language*

SUMÁRIO

1 INTRODUÇÃO

Segundo Rice (2009, p. 28-31), ter um sistema de qualidade é em geral um dos maiores desafios encontrados pelas empresas de software, devido à alta complexidade dos produtos, entre outras dificuldades encontradas no processo de desenvolvimento.

Os sistemas devem fazer corretamente o que o cliente requisitou, com eficiência, agilidade e segurança. Para garantir a qualidade dos sistemas de software são utilizados os testes manuais e testes automatizados, os quais asseguram tais características do software.

Testes automatizados são, na verdade, *scripts* que, uma vez escritos, podem validar rapidamente diversos casos de teste. Geralmente são utilizados em softwares que possuem uma vida longa no mercado, com constantes implementações, melhorias e correções, as quais podem gerar problemas nas funcionalidades que antes executavam perfeitamente.

> O teste automatizado aumenta a produtividade e atinge em tempo menor aquilo que é em geral rotineiro no teste. O teste manual não pode ser eliminado; deve, sim, ser reduzido o máximo possível e focado naquilo que é muito cara [sic] automatizar. (MOLINARI, 2010 p. 37).

Segundo Caetano (2007), faz parte da automação de testes de software, no que se refere ao ciclo de desenvolvimento, somente as ferramentas que automatizam a execução dos testes em si. O *TestComplete* é uma das ferramentas utilizadas para a geração de testes automatizados. Uma das suas funcionalidades é o detalhamento de cada execução através do *Test Log.*

Conforme Smartbear (2013b), o *Test Log* apresenta o resultado final de cada execução, com um resumo com as principais informações do teste, tais como, o horário de início e final, o tempo total, a quantidade de erros, alertas e a quantidade de itens que executaram com sucesso. O detalhamento do *Test Log* apresenta cada passo da execução.

Mesmo que o *TestComplete* seja uma ferramenta bastante completa, encontram-se algumas dificuldades para analisar e comparar vários registros de *log* de testes manualmente.

Diante do exposto, este trabalho apresenta uma solução para melhorar a análise e comparação dos *Test Logs.* Através desta ferramenta será possível importar os *Test Logs* do *TestComplete*, a qual permitirá visualizar várias execuções e os projetos dos seus respectivos responsáveis, com a possibilidade de gerar gráficos que auxiliem em uma análise mais ágil e eficiente.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho é desenvolver uma ferramenta que auxilie a análise e a comparação de *logs* gerados pelas execuções dos testes através da ferramenta *TestComplete*.

Os objetivos específicos do trabalho são:

a) fornecer ao gestor da qualidade uma ferramenta para auxiliar a gestão da sua equipe de testes;
b) conhecer os detalhes dos *logs* de testes, seu padrão e suas informações geradas;
c) prover ao testador uma ferramenta para comparação de distintas execuções dos testes;
d) permitir ao gestor da qualidade e testador compor uma base histórica dos testes comparados.
e) diminuir o tempo gasto com análise e comparação dos *Test Logs*.

1.2 ESTRUTURA DO TRABALHO

No primeiro capítulo tem-se a introdução ao tema principal deste trabalho com a apresentação da justificativa e dos objetivos.

No segundo capítulo apresenta-se a fundamentação teórica pesquisada sobre área de testes, automação de testes, a ferramenta *TestComplete*, *TestLogs* além de trabalhos correlatos.

O terceiro capítulo apresenta o desenvolvimento da ferramenta para análise de *logs* do *TestComplete*, iniciando-se com o levantamento de informações, tendo na sequência as técnicas e ferramentas utilizadas. Também são apresentados alguns diagramas para auxiliar na compreensão do sistema, a sua operacionalidade, resultados obtidos e discussões.

No quarto capítulo têm-se as conclusões deste trabalho bem como são apresentadas sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os assuntos, apresentados nas seções a seguir, sobre área de testes, automação de testes, a ferramenta *TestComplete, Test Logs* e trabalhos correlatos.

2.1 ÁREA DE TESTES

Segundo Bastos, Rios, Cristalli e Moreira (2007, p. 17), o teste, da maneira como é executado pela maioria das empresas, é realizado dentro do processo de desenvolvimento. Em geral são executados pelos próprios desenvolvedores ou por usuários do sistema. Um software criado dessa maneira, raramente estará livre de defeitos, pois esse processo de teste serve apenas para garantir que as especificações ou requisitos foram implementados.

Para evitar esses defeitos, é preciso criar uma área de testes, definir os processos de teste, com metodologia própria, minimizando os riscos causados por defeitos no processo de desenvolvimento. Na área de testes é preciso uma equipe com especialistas treinados para isso e não obrigar o analista a executar essa atividade.

> Nem os analistas de sistemas, nem os usuários são técnicos de teste de software. Os testes para serem feitos corretamente [sic], é necessário [sic] que os testes sejam executados por profissionais capacitados, usando metodologia apropriada, em ambiente adequado e, muitas vezes, tendo à disposição ferramentas de automação. (BASTOS; RIOS; CRISTALLI; MOREIRA, 2007, p. 18).

Ainda segundo Bastos, Rios, Cristalli e Moreira (2007, p.19), quando os testes são tratados como um processo organizado e integrado ao processo de desenvolvimento, consegue-se reduzir os custos de manutenção consideravelmente. Então, pode-se dizer que investir na área de testes pode ser caro, mas os seus benefícios são essenciais para o desenvolvimento do software.

2.2 AUTOMAÇÃO DE TESTES

Segundo Hayes (2009, p. 15), é notável uma evolução das ferramentas de automação

de testes nos últimos 25 anos. Isto faz com que a cada dia a importância de se investir em testes aumente gradativamente, na tentativa de reduzir custos com problemas e também para que os softwares atendam às expectativas e necessidades dos clientes. Ainda segundo Hayes (2009, p. 15), mesmo com toda essa evolução, poucas empresas adotam ainda a automação de testes nos seus processos.

Segundo Dustin (2015), hoje um dos principais motivos para se utilizar a automação de testes é a exigência maior na qualidade do produto entregue, pois automatizando-se o processo de testes consegue-se realizar mais testes em menos tempo. Ainda segundo Dustin (2015), automação de testes pode ser definida como a utilização de uma ferramenta para criar *scripts* de testes e executá-los. Ao final deste processo, comparam-se os resultados obtidos em cada execução com os resultados.

Segundo Graham e Fewster (1999, p. 4), existem uma diferença clara entre testes e automação de testes. Alguns confundem testes e automação de testes, mas há uma diferença significativa entre os dois, sendo que no primeiro se realiza a tarefa de testar e no segundo é utilizada uma ferramenta capaz de imitar a interação com a aplicação tal como um ser humano faria, ou seja, a ferramenta automatizará o teste que deseja ser realizado.

> Automação de teste tem uma história de fracasso. Apesar da explosão dos softwares em cada aspecto de nossas vidas, apenas uma pequena percentagem dos testes é realmente automatizada hoje. Assim, analisar como as ferramentas de automação de teste têm evoluído é útil somente se isto ajuda a compreender para onde vamos a partir daqui. Para tal, você precisa entender as tecnologias e as forças do mercado que as moldaram, por que isto aconteceu e o que isso prenuncia para o futuro. (HAYES, 2009, p. 15, tradução nossa).

Segundo Molinari (2010, p. 35), ao investir em testes, investe-se em prevenção de defeitos. Investir na automação de testes é obter maior satisfação do usuário do software. A imagem da empresa melhora no quesito software com qualidade. As dúvidas e incertezas em relação a problemas ou defeitos que cercam o software diminuem e também reduz-se o custo de manutenção de um software em produção.

Existem diversos desafios na automação de testes. Segundo Rice (2009, p. 28-31), as pessoas concordam com dois grandes desafios de automação de teste: superar as expectativas irrealistas e conseguir apoio à gestão para automação de teste. As expectativas irreais estão na questão de se acreditar que com a automação irão se resolver os problemas existentes de imediato ou até mesmo que em apenas uma única solução de automação irá ser possível resolver tudo, e é nesse ponto que se percebe que não é tão simples o processo de automatizar os testes, mas que não é algo intangível, apenas trabalhoso. Obter suporte de gerenciamento da automação de teste é definir boas práticas de uso da ferramenta e também ter um padrão de

uso bem especificado, tendo assim uma transparência no processo de automação de testes.

A definição de automação de testes, segundo Bartié (2002, p. 63), é a utilização de ferramentas de testes que simulem usuários ou atividades humanas de forma a não empregar procedimentos manuais no processo de execução dos testes. Isso requer profissionais especializados, exige mais detalhes no planejamento e tempo adicional para o desenvolvimento e automação dos testes.

Conforme Molinari (2010, p. 39), alguns dos principais conceitos de automação são:

a) gravador ou *recorder*: diversas ferramentas de testes possuem a gravação, que quando acionada irá registrar todas as ações do usuário e, no final, gravar um *script* de teste;
b) *script* de teste: um conjunto de ações que a ferramenta de automação de testes irá executar;
c) executor de teste ou *playback*: é o recurso da ferramenta que irá executar o *script* de teste que foi gravado;
d) ponto de verificação ou *checkpoints*: ação que irá verificar se o caso de teste do *script* de teste foi validado com sucesso;
e) tempo de pensar ou *think time*: um atraso temporal que simula o tempo de interação das ações do usuário;
f) conceito de programação: depuração dos *scripts*, que exige lógica de programação.

2.3 A FERRAMENTA *TESTCOMPLETE*

Conforme Smartbear (2013a), o *TestComplete* é uma das ferramentas com baixo custo para automação de testes. Desenvolvida pela AutomatedQA, a ferramenta oferece um amplo conjunto de funcionalidades:

a) suporte nativo para automação de sistemas desenvolvidos em diversas tecnologias, tais como Java, .NET, Visual Basic, Borland Delphi, WEB, entre outras;
b) gravar testes automatizados através do recurso *point-and-click Automated Test Recording Engine*;
c) editor de código para visualizar e gerir scripts de teste automatizados, com função de autocompletar;
d) test logs gera registros detalhados de todas as ações realizadas durante o teste

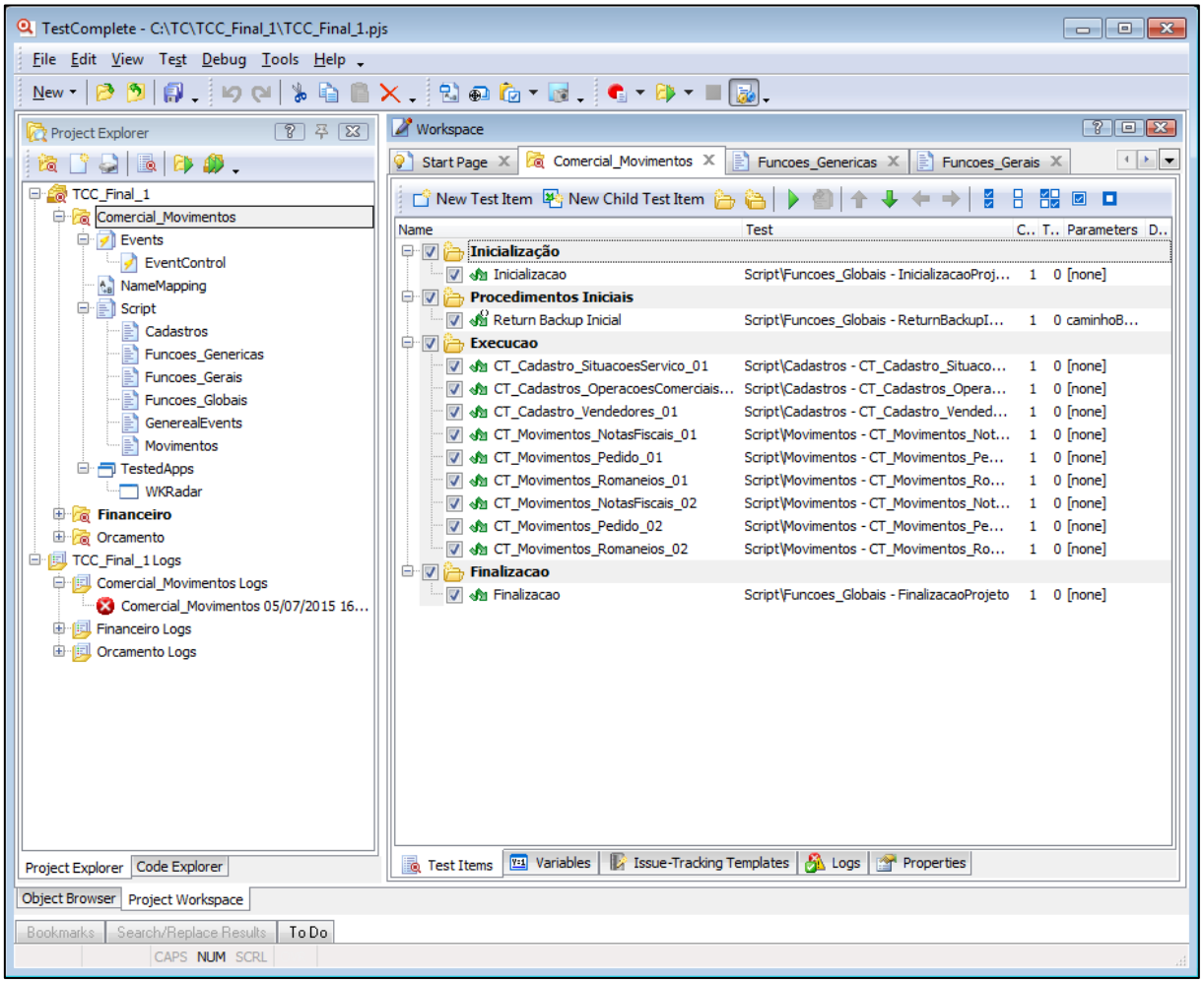
automatizado, mais detalhes dos *Test Logs* são apresentados na seção 2.4.

Segundo Mota (2011), o *TestComplete* possui uma grande flexibilidade e consegue executar uma grande lista de testes, tais como testes funcionais, testes de unidade, testes de carga, testes de regressão e testes em modo distribuído ou cliente/servidor.

Ainda segundo Mota (2011), para a criação de *scripts* a ferramenta requer certa experiência em lógica de programação e, claro, em técnicas de automação de testes. Como a ferramenta dispõe de muitas funcionalidades, é essencial um bom treinamento para sua utilização da forma correta.

A seguir na Figura 1 é apresentado um *print screen* da ferramenta *TestComplete*.

Figura 1 – Ferramenta *TestComplete*



2.4 *TEST LOGS*

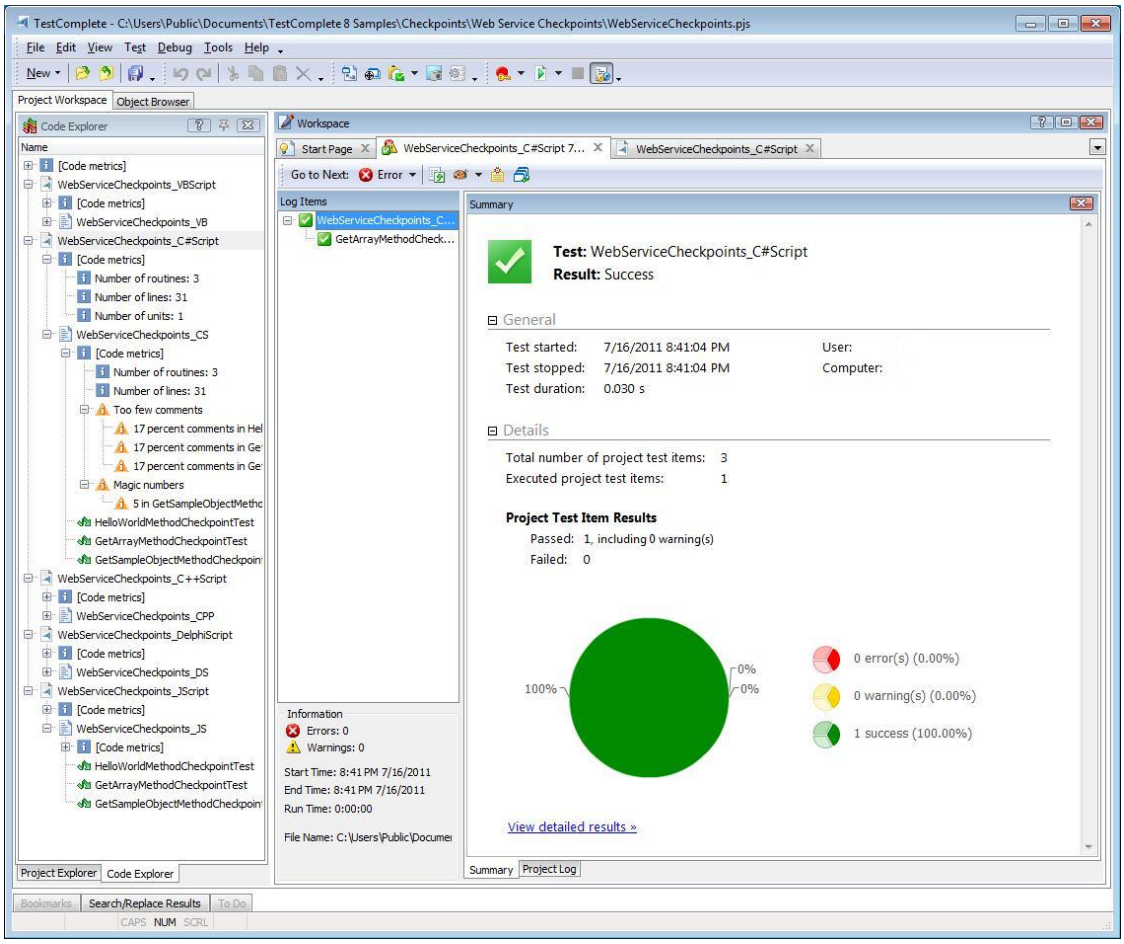
Conforme Smartbear (2013b), no final de cada execução dos testes automatizados o *TestComplete* gera *logs* de testes detalhando cada ação realizada. Os *Test Logs* podem ser visualizados diretamente na ferramenta ou podem ser exportados no formato HTML, XML ou MHT para que possa ser visualizado em computadores que não possuem o *TestComplete* instalado.

Segundo Smartbear (2013b), os *logs* de teste permitem que:

a) veja-se *screenshots* do seu aplicativo testado para cada etapa de teste e compará-los lado a lado com os capturados durante a gravação de teste para identificar rapidamente mudanças críticas na aplicação;
b) adicione-se texto personalizado e conteúdo HTML para os *logs* e cores nos códigos, para uma melhor análise visual;
c) capture-se automaticamente imagens de *desktop*;
d) filtre-se mensagens de *log* pelo seu texto, tipo, prioridade, tempo e outros atributos;
e) reveja-se os resumos dos registros de um teste automatizado;
f) visualize-se a impressão e os registros de impressão;
g) exporte-se os dados de *log* para arquivos.

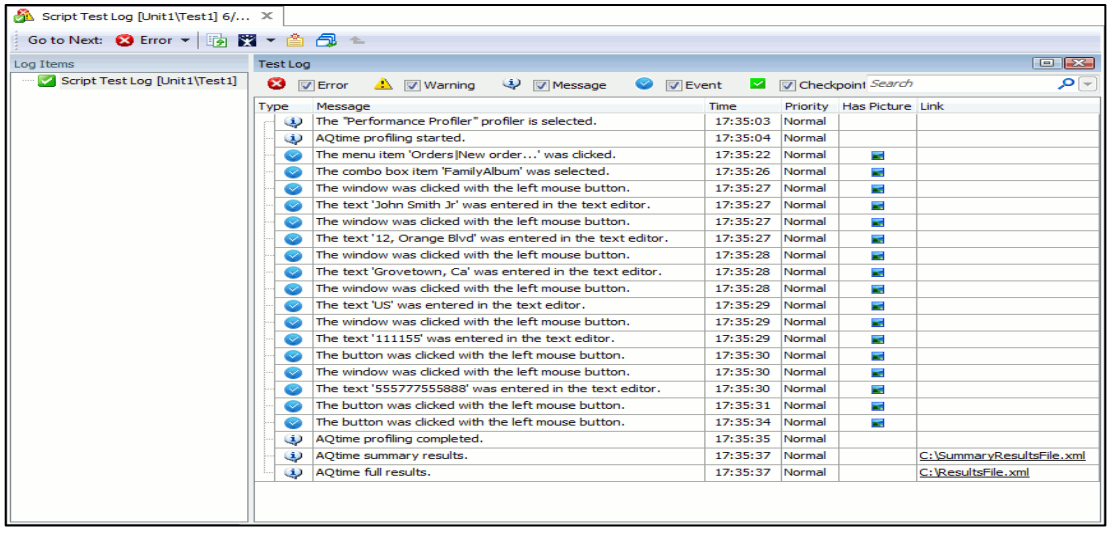
Na Figura 2 é apresentado um *print screen* de um resumo do *Test Log* no final da execução de um teste automatizado.

Figura 2 – Resumo do *Test Log*



Na Figura 3 é apresentado um *print screen* do *Test Log* detalhado de cada ação executada nos testes automatizados.

Figura 3 – *Test Log* detalhado



2.5 TRABALHOS CORRELATOS

Pode-se citar como trabalhos correlatos as monografias realizadas pela aluna Adriana Fronza Marcos e pelo aluno Douglas de Oliveira Waltrick para conclusão do curso de Ciências da Computação e Sistemas de Informação na Universidade Regional de Blumenau.

O trabalho de Marcos (2007) foi desenvolver uma ferramenta para apoio a automatização de testes. A ferramenta utiliza os arquivos de programas desenvolvidos em *Delphi*, ou seja, analisa as telas geradas e gera arquivos de testes na linguagem *DelphiScript*, servindo de entrada para a ferramenta de automatização de testes *TestComplete*. São utilizados templates para formatação dos *scripts* de testes, buscando tornar a geração mais flexível.

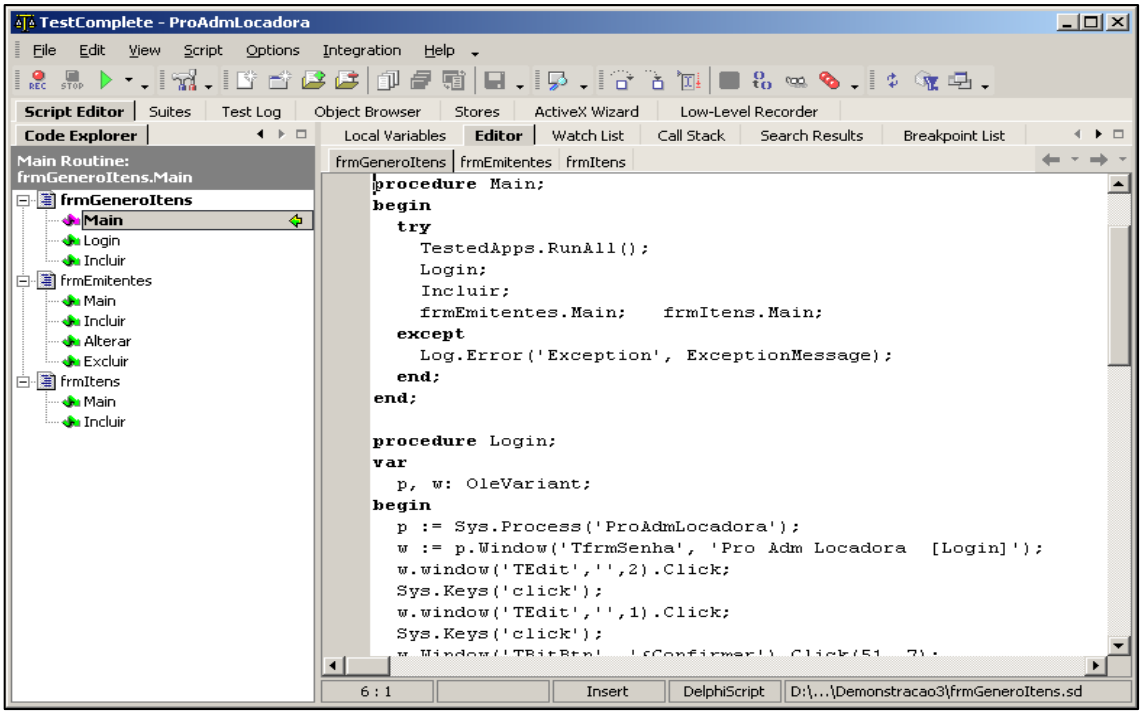
Na Figura 4 é apresentada a tela desenvolvida em Delphi, que será analisada para a geração dos *scripts* de testes.

Figura 4 – Interface para identificação de componentes

Fonte: Marcos (2007).

Na Figura 5 é apresentado o *script* de testes gerado na ferramenta *TestComplete.*

Figura 5 – *Scripts* de teste na ferramenta TestComplete

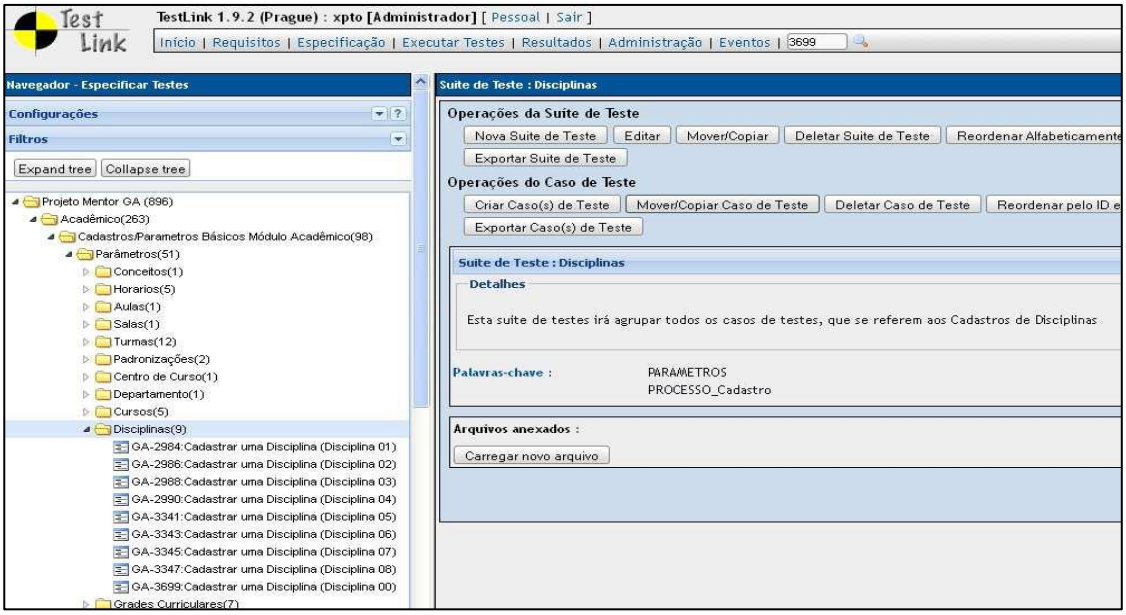


Fonte: Marcos (2007).

O trabalho de Waltrick (2011) foi desenvolver um *plugin* dentro do *framework TestComplete* que integra-se com a ferramenta CASE *TestLink*, com o objetivo de manter as duas ferramentas síncronas, deixando mais produtiva a atividade do analista de teste de codificar os *scripts* e os resultados referentes à execução dos testes automatizados.

Na Figura 6 é apresentada a ferramenta *TestLink* onde são criados os casos de testes.

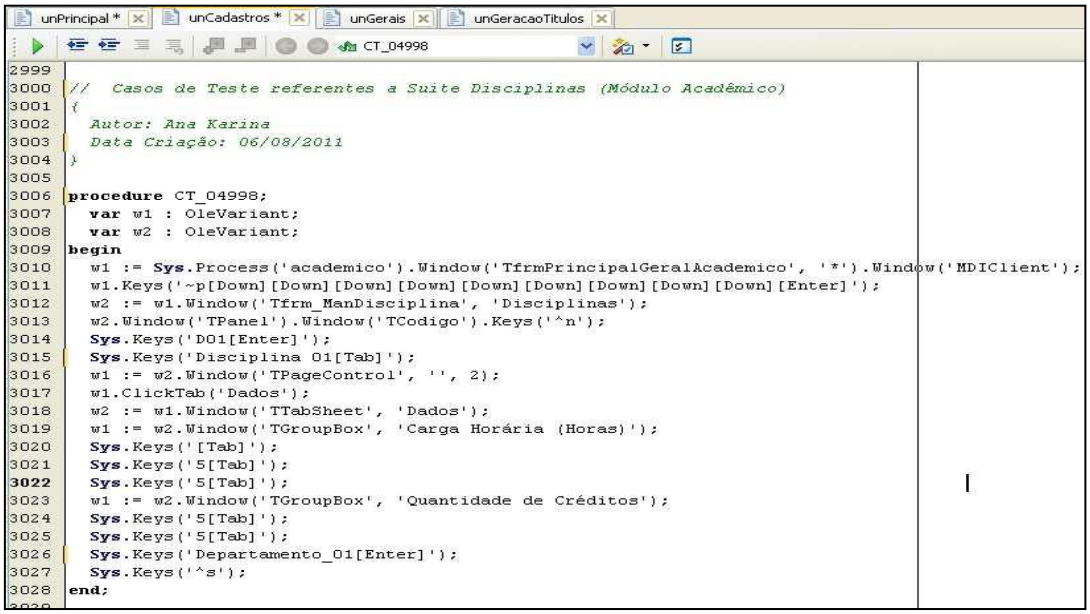
Figura 6 – Tela de criação de casos de teste



Fonte: Waltrick (2011).

Na Figura 7 é apresentado o código gerado para o *script* de automatização dos passos do caso de teste.

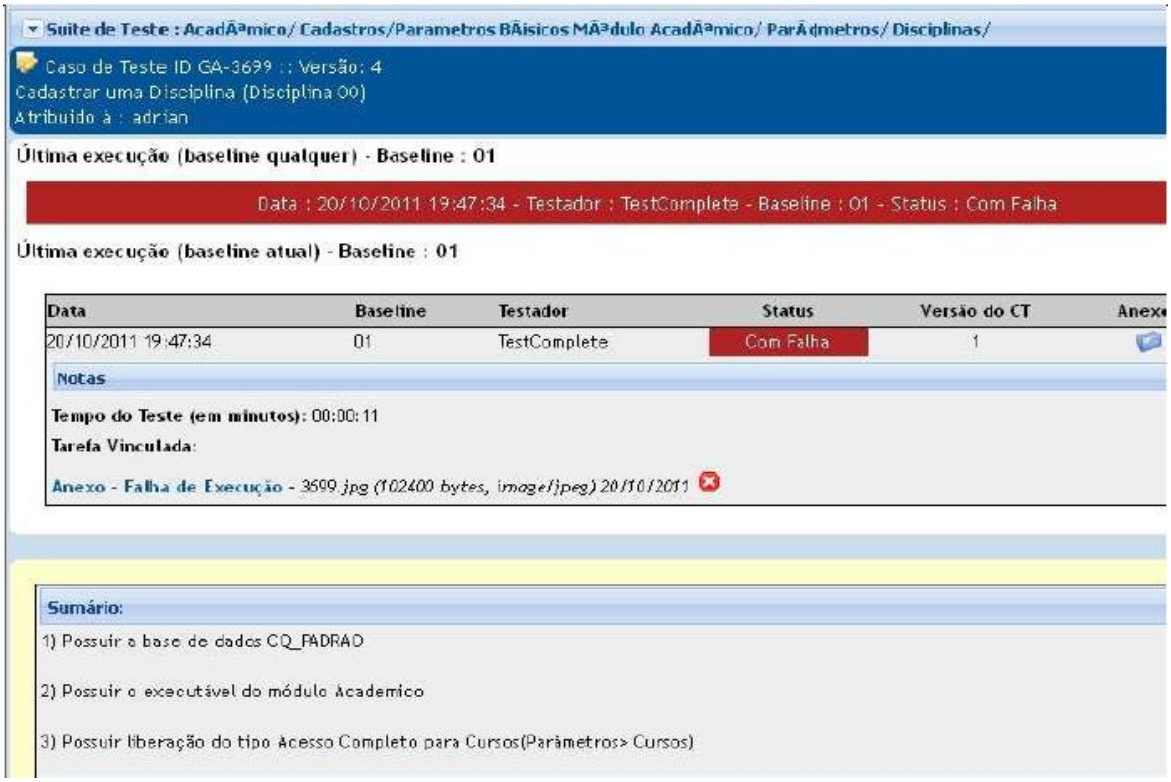
Figura 7 – Código fonte do *script* de automatização



Fonte:Waltrick (2011).

Na Figura 8 é apresentado o registro automático na ferramenta *TestLink*.

Figura 8 – Registro automático na ferramenta TestLink



Fonte: Waltrick (2011).

Os trabalhos correlatos apresentados automatizaram um processo realizado por usuários da ferramenta *TestComplete*, e este é o objetivo da ferramenta que será criada neste trabalho: automatizar a análise e comparação de *logs* gerados pelas execuções dos testes através da ferramenta *TestComplete*.

3 DESENVOLVIMENTO DA FERRAMENTA

Neste capítulo estão descritas as particularidades técnicas, tais como a sua descrição e a apresentação dos requisitos funcionais e não funcionais, diagrama de casos de uso e suas descrições, diagrama de entidade relacionamento, a sua implementação, as principais ferramentas utilizadas, a sua operacionalidade e, também os resultados e discussões.

3.1 LEVANTAMENTO DE INFORMAÇÕES

A ferramenta desenvolvida fundamenta-se em uma plataforma desktop para auxiliar na visualização dos projetos e seus respectivos *Test Logs* gerados após as execuções dos testes automatizados.

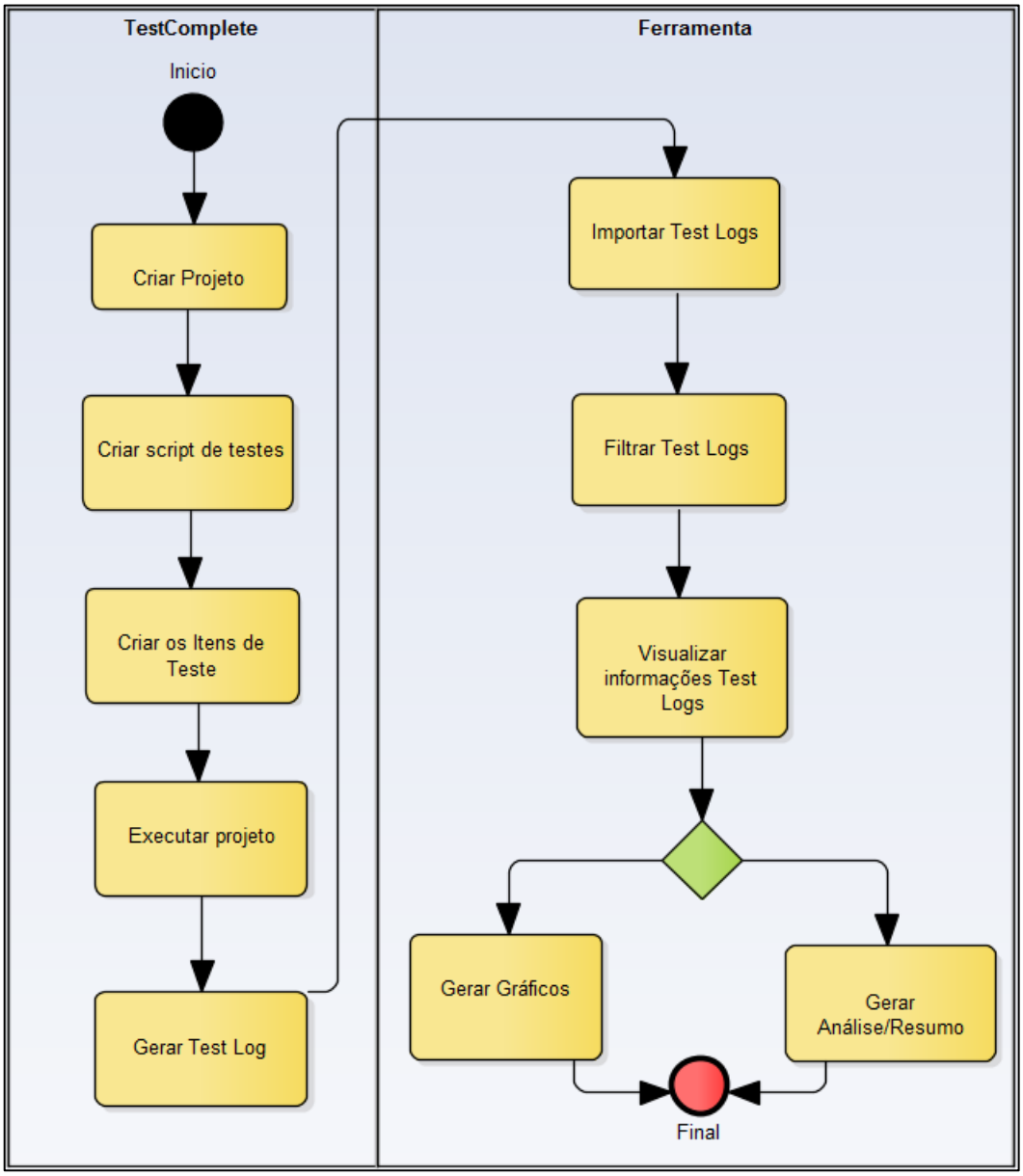
Com base nas necessidades básicas do usuário da ferramenta *TestComplete*, constatou-se uma dificuldade em analisar os resultados dos testes. A ferramenta permite importar os *Test Logs* gerados, obtendo assim uma visualização ampla de vários resultados (*Test Logs*), e não de apenas um *Test Log* por vez.

A ferramenta ainda permite a geração de gráficos que auxiliam na análise e comparação de vários *Test Logs.* Para geração dos gráficos devem-se importar os *Test Logs* e em seguida filtrar os dados importados para a tela principal da ferramenta.

Na criação dos projetos na ferramenta *TestComplete* é necessário incluir os itens de teste iniciando com a sigla “CT_”. Esse padrão foi necessário para identificar quando o item que esta sendo executado é de fato um caso de teste.

O processo de funcionamento da ferramenta, em alto nível, pode ser visualizado na Figura 9.

Figura 9 – Funcionamento da Ferramenta, em alto nível



3.2 ESPECIFICAÇÃO

Nesta seção são apresentados os principais requisitos funcionais e não funcionais, sua rastreabilidade com casos de uso e o MER.

3.2.1 Requisitos do Sistema

O Quadro 1 apresenta os requisitos funcionais da ferramenta e sua rastreabilidade, ou seja, vinculação com o(s) caso(s) de uso associado(s). Optou-se por ter um caso de uso para cada gráfico gerado.

Quadro 1 – Requisitos funcionais

Requisitos Funcionais	**Caso de Uso**
RF01. A aplicação deve permitir ao gestor da qualidade/testador informar o caminho onde se encontram os *Test Logs* a serem importados.	UC01
RF02. A aplicação deve permitir ao gestor da qualidade/testador gerar uma análise dos *Test Logs*.	UC02
RF03. A aplicação deve permitir ao gestor da qualidade/testador emitir o gráfico com a quantidade de casos de teste por projeto.	UC03
RF04: A aplicação deve permitir ao gestor da qualidade/testador emitir o gráfico com a quantidade de casos de teste por responsável.	UC04
RF05: A aplicação deve permitir ao gestor da qualidade/testador emitir o gráfico com a quantidade total de execuções dos casos de teste.	UC05
RF06: A aplicação deve permitir ao gestor da qualidade/testador emitir o gráfico com a quantidade total de execuções dos casos de teste com erro.	UC06
RF07. A aplicação deve permitir ao gestor da qualidade/testador emitir o gráfico com a quantidade de erros e alertas dos projetos.	UC07
RF08. A aplicação deve permitir ao gestor da qualidade/testador emitir o gráfico com o tempo total de testes dos projetos.	UC08
RF09. A aplicação deve permitir ao gestor da qualidade/testador emitir o gráfico com a quantidade de erros e alertas por responsável.	UC09
RF10. A aplicação deve permitir ao gestor da qualidade/testador emitir o gráfico com o tempo total de testes por responsável.	UC10
RF11. A aplicação deve permitir ao gestor da qualidade/testador comparar dois ou mais *Test Logs* do mesmo projeto.	UC11
RF12. A aplicação deve permitir ao gestor da qualidade/testador emitir o gráfico com o tempo total de cada caso de teste.	UC12
RF13. A aplicação deve permitir ao gestor da qualidade/testador emitir o	UC13

gráfico com a quantidade de erros de cada caso de teste.	
RF14. A aplicação deve permitir ao gestor da qualidade/testador emitir o gráfico com a quantidade de alertas de cada caso de teste.	UC14

O Quadro 2 lista os requisitos não funcionais do sistema.

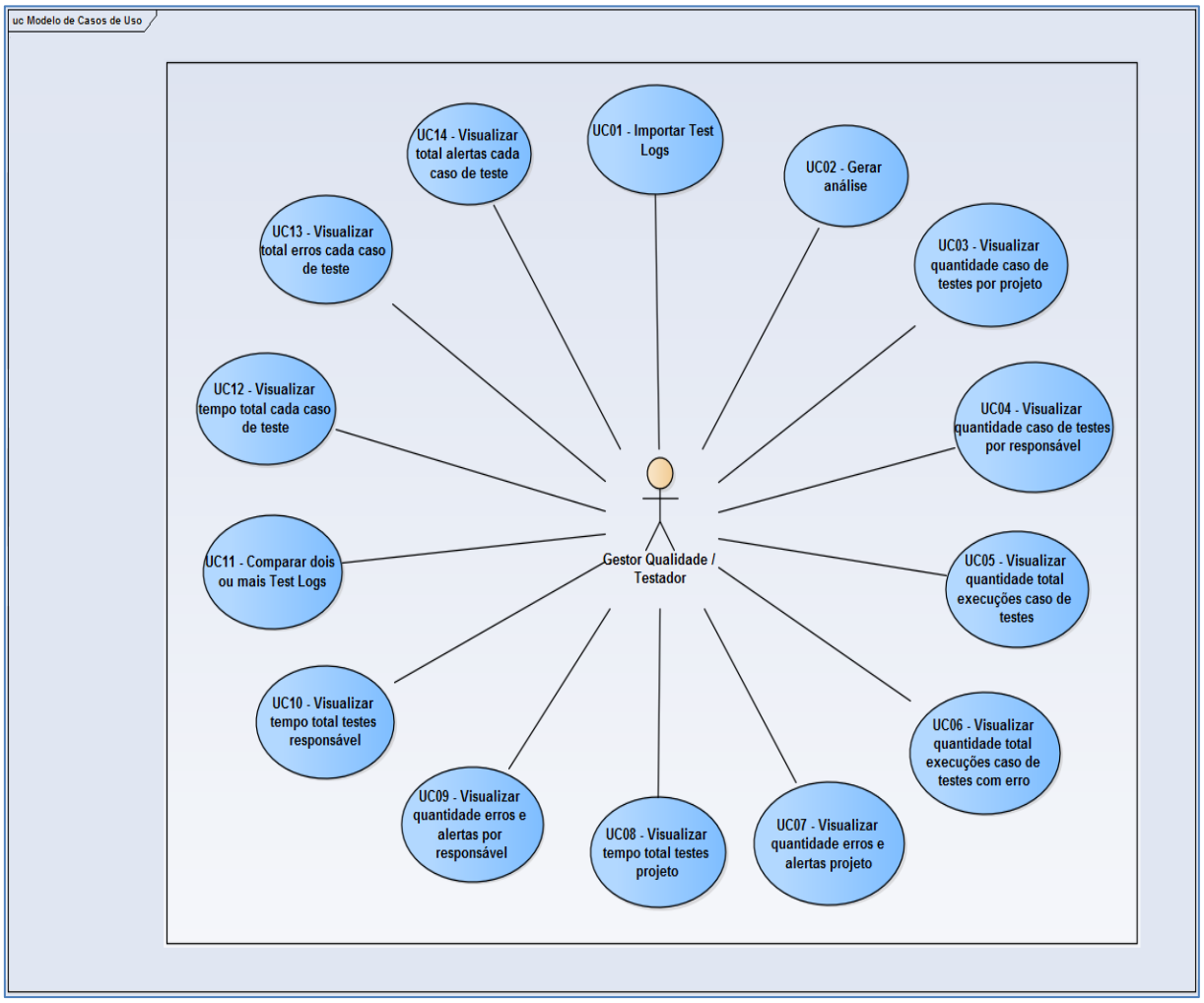
Quadro 2 – Requisitos não funcionais

Requisitos Não Funcionais
RNF01. O ambiente de desenvolvimento a ser utilizado será Netbeans.
RNF02. A ferramenta deverá ser desenvolvida na linguagem Java para plataforma *desktop*.
RNF03. Para análise dos *Test Logs* será utilizado o método de pesquisa em arquivos por expressões regulares.
RNF04: A ferramenta deverá utilizar banco de dados MySQL.
RNF05. O ambiente para geração dos gráficos será o *framework JFreeChart*.

3.2.2 Diagrama de Casos de Uso

A Figura 10 demonstra o diagrama de casos de uso da ferramenta. A descrição detalhada dos casos de uso é apresentada no Apêndice A.

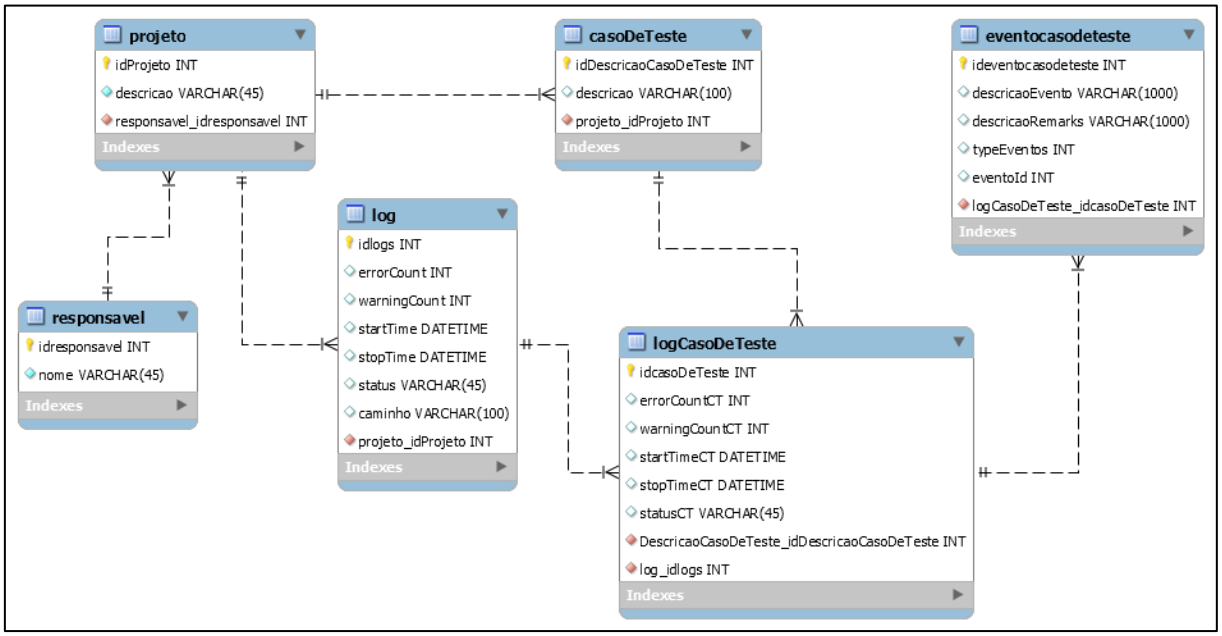
Figura 10 – Diagrama de casos de uso



3.2.3 Modelo Entidade Relacionamento

A Figura 11 contempla o MER do sistema. O dicionário de dados é apresentado no Apêndice B.

Figura 11 – Modelo entidade relacionamento



Uma breve descrição das entidades criadas para o desenvolvimento do sistema é a seguinte:

a) projeto – entidade que armazena os dados referentes aos projetos;
b) responsavel – entidade que armazena os dados referentes aos responsáveis;
c) casoDeTeste – entidade que armazena os dados referentes aos caso de testes;
d) log – entidade que armazena os dados referentes aos *logs*;
e) logCasoDeTeste – entidade que armazena os dados referentes aos casos de testes dos *logs*;
f) eventocasodeteste – entidade que armazena os dados referentes aos erros e alertas dos caso de testes executados.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Foram utilizadas as seguintes ferramentas para a construção da ferramenta:

a) Netbeans 8.0, como plataforma de desenvolvimento;
b) ferramenta MySQL Workbench, para construção do Modelo de Entidade e Relacionamento (MER);
c) ferramenta ArgoUML para a modelagem do diagrama de casos de uso;
d) MySQL, como banco de dados.

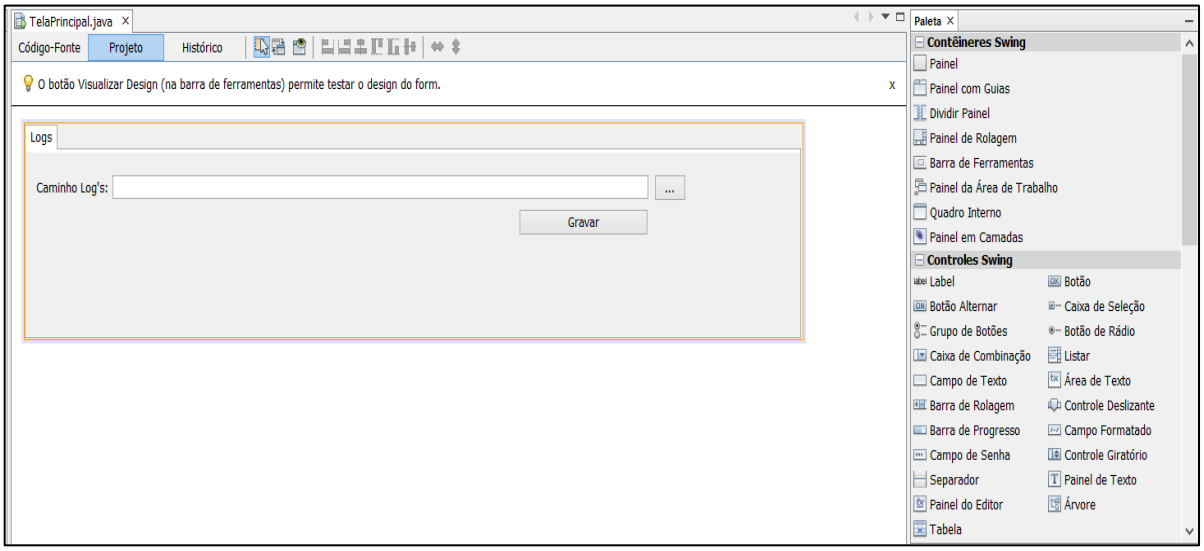
Os dados dos arquivos *Test Logs* são importados utilizando o método de pesquisa em arquivos por expressões regulares, que provê uma forma concisa e flexível de identificar cadeias de caracteres de interesse e são armazenadas no banco de dados MySQL. O MySQL é um Sistema Gerenciador de Bando de Dados (SGDB) que utiliza a Linguagem de Consulta Estruturada (*Structured Query Language*, SQL).

Para a geração dos gráficos foi utilizado o *framework JFreeChart*, que permite a criação de gráficos tanto interativos quanto não interativos. O *JFreeChart* desenha automaticamente as escalas dos eixos e legendas.

3.3.1.1 Netbeans IDE 8.0.2 e Java

O Netbeans IDE 8.0.2 oferece facilidades para criação de interfaces, para criação das interfaces da ferramenta foi utilizado o construtor *Graphical User Interface* (GUI). Na Figura 12 é apresentado um *print screen* da GUI que auxilia a criação das interfaces gráficas.

Figura 12 – GUI para criação de interfaces gráficas



A linguagem de programação Java foi utilizada em todo o desenvolvimento da ferramenta. Na Figura 13 demonstra-se a um *print screen* classe Java `Projeto` que recebe os dados dos projetos encontrados nos *Test Logs* importados.

Figura 13 – Classe Java **`Projeto`** que recebe dados dos projetos

TelaPrincipal.java | Projeto.java

Código-Fonte | Histórico

```
7
8   /**
9    *
10   * @author Junior
11   */
12  public class Projeto {
13      private String descricaoProjeto;
14      private int codigo;
15      private int codResponsavel;
16
17      public int getCodigo() {
18          return codigo;
19      }
20
21      public void setCodigo(int codigo) {
22          this.codigo = codigo;
23      }
24
25      public int getCodResponsavel() {
26          return codResponsavel;
27      }
28
29      public void setCodResponsavel(int codResponsavel) {
30          this.codResponsavel = codResponsavel;
31      }
32
33      public String getDescricaoProjeto() {
34          return descricaoProjeto;
35      }
36
37      public void setDescricaoProjeto(String descricaoProjeto) {
38          this.descricaoProjeto = descricaoProjeto;
39      }
40
41
42
43  }
44
```

3.3.1.2 Expressões regulares

As expressões regulares provêm uma forma de identificar uma cadeia de caracteres, através de uma composição de símbolos e caracteres com funções especiais que, agrupados entre si, formam uma sequência, uma expressão. A expressão regular foi utilizada no momento da leitura dos arquivos *Test Logs* para identificar as informações de interesse.

Em Java as expressões regulares são conhecidas como *Regular Expressions* (*Regex*). Para utilizá-las é necessário importar o pacote de classes java.util.regex (DEVMEDIA, 2007).

Na Figura 14 demonstra-se um *print screen* da utilização do *Regex* no método de leitura de um arquivo de *Test Logs*.

Figura 14 – Utilização *Regex* na leitura do arquivo

TelaPrincipal.java | Projeto.java | LerArquivo.java

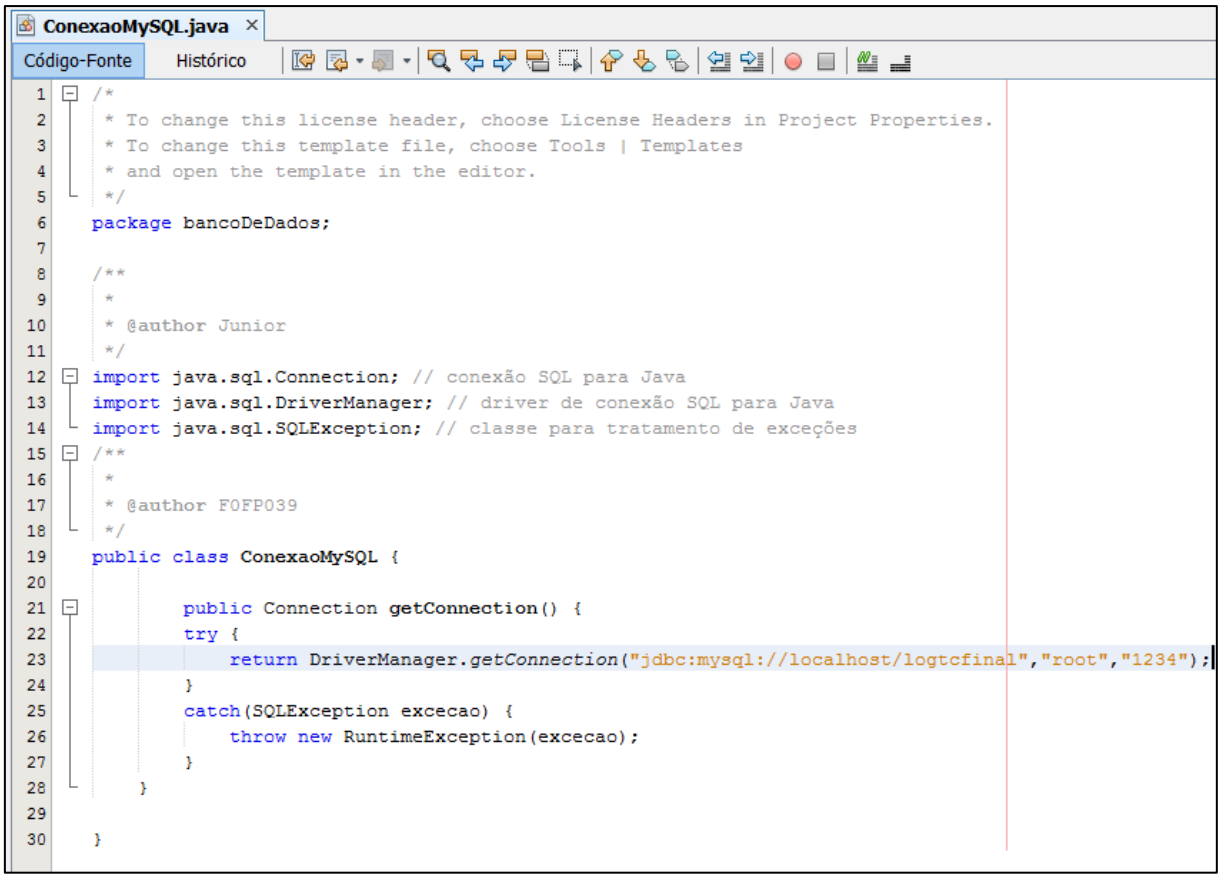
Código-Fonte | Histórico

```
25  public String lerArquivo(Reader input, String inicioPesquisa, String finalPesquisa) throws IOException {
26
27      BufferedReader rd = new BufferedReader(input);
28      String name = null;
29      boolean achouCT = false;
30      START_RE = inicioPesquisa;
31      END_RE = finalPesquisa;
32      String conteudoEncontrado = "";
33      Pattern startPattern = Pattern.compile(START_RE);
34
35      String line;
36      while ((line = rd.readLine()) != null) {
37
38          if (line.trim().isEmpty()) { // ignore linha em branco
39              continue;
40          }
41          if (name == null) { // já achou o inicio do <Node?
42              Matcher start = startPattern.matcher(line); // procure inicio do Node
43              if (start.matches()) {
44
45                  name = start.group();
46              }
47          } else {
48              if (line.matches(END_RE)) {
49                  // fim do nome (linha anterior)
50                  conteudoEncontrado = name;
51                  name = null;  // procure proximo
52              } else {
53
54                  name += line;
55              }
56          }
57      }
58      return conteudoEncontrado;
59  }
```

3.3.1.3 MySQL

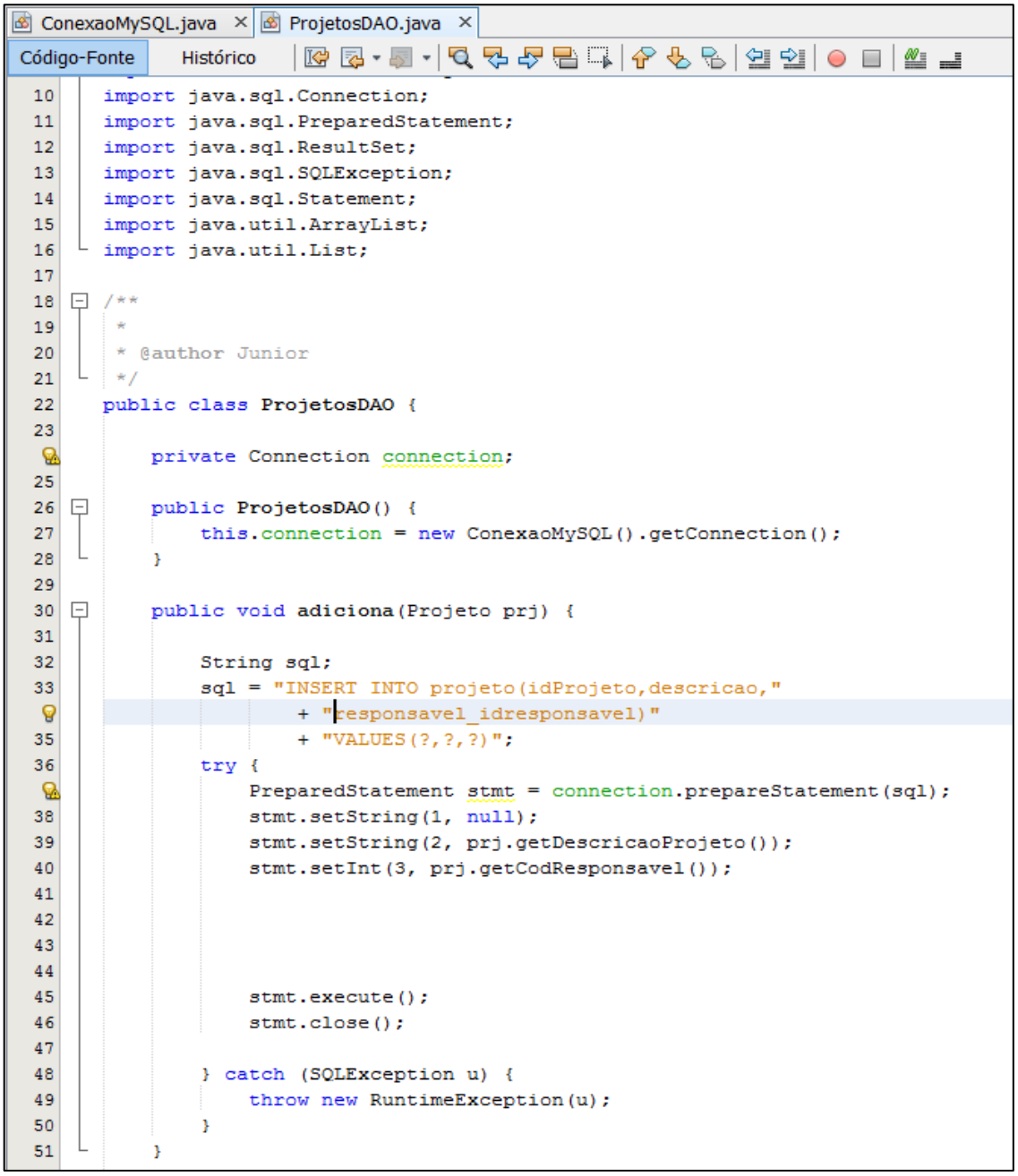
O *MySQL* é um sistema gerenciador de banco de dados relacional. Na Figura 15 demonstra-se um *print screen* da classe Java que faz a conexão com o banco de dados MySQL.

Figura 15 – Classe Java de conexão com o banco de dados



Para realizar os comandos de inserção e consulta no banco de dados, foram desenvolvidas classes específicas para realizar estas funções. Na Figura 16 demonstra-se um *print screen* da implementação da classe `ProjetosDAO` com a função de inserir os dados importados dos *Test Logs* para a entidade Projeto.

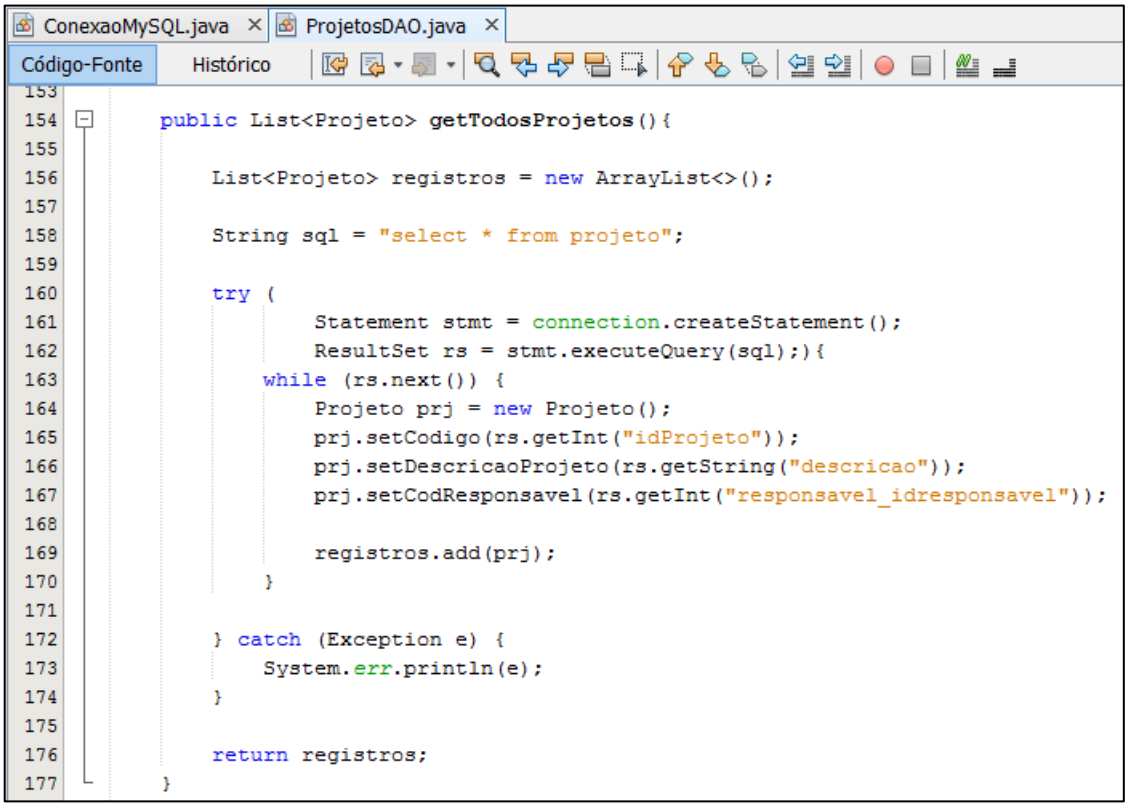
Figura 16 – Implementação da classe **`ProjetoDAO`**




```
10  import java.sql.Connection;
11  import java.sql.PreparedStatement;
12  import java.sql.ResultSet;
13  import java.sql.SQLException;
14  import java.sql.Statement;
15  import java.util.ArrayList;
16  import java.util.List;
17
18  /**
19   *
20   * @author Junior
21   */
22  public class ProjetosDAO {
23
25
26      public ProjetosDAO() {
27          this.connection = new ConexaoMySQL().getConnection();
28      }
29
30      public void adiciona(Projeto prj) {
31
32          String sql;
33          sql = "INSERT INTO projeto(idProjeto,descricao,"
35              + "responsavel_idresponsavel)"
36              + "VALUES(?,?,?)";
37          try {
38              PreparedStatement stmt = connection.prepareStatement(sql);
39              stmt.setString(1, null);
40              stmt.setString(2, prj.getDescricaoProjeto());
41              stmt.setInt(3, prj.getCodResponsavel());
42
43
44
45              stmt.execute();
46              stmt.close();
47
48          } catch (SQLException u) {
49              throw new RuntimeException(u);
50          }
51      }
```

Na Figura 17, demonstra-se um *print screen* da função que realiza uma consulta na entidade Projeto.

Figura 17 – Função de consulta da entidade Projeto



3.3.1.4 Arquivos *Test Logs*

No final de cada execução dos testes automatizados, o *TestComplete* gera arquivos de *logs* dos testes, detalhando cada ação realizada. Esses arquivos são gerados no formato *Description.TcLog*. Só se consegue visualizá-los abrindo-os na ferramenta *TestComplete*.

Os arquivos de *Test Logs* seguem um padrão, com *tags* específicas para identificar cada informação. Na Figura 18 demonstra-se um *print screen* do padrão de um arquivo *Test Logs*.

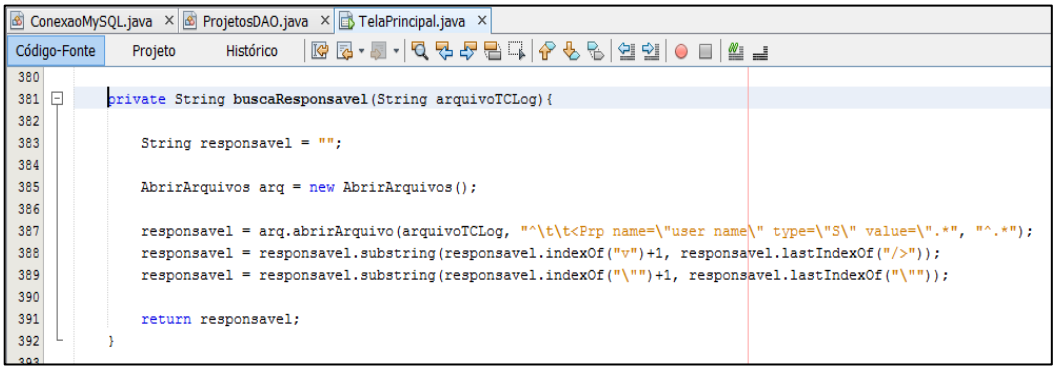
Figura 18 – Padrão de um arquivo *Test Logs*

Description.tcLog

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE Nodes [

<!ENTITY % NameValue "CDATA">
<!ENTITY % PropType "(I|S|D|L|H|B)">
<!ENTITY % VersionValue "CDATA">

<!ELEMENT Prp (#PCDATA)>
<!ELEMENT Node (Node|Prp)*>
<!ELEMENT Nodes (Node)+>

<!ATTLIST Nodes version CDATA "1">

<!ATTLIST Node name CDATA #REQUIRED>

<!ATTLIST Prp name CDATA #REQUIRED>
<!ATTLIST Prp type CDATA #REQUIRED>
<!ATTLIST Prp value CDATA #REQUIRED>
]>
<Nodes version="1">
    <Node name="root">
        <Prp name="start time" type="D" value="42156.935309375"/>
        <Prp name="test type" type="S" value="Orcamento"/>
        <Prp name="stop time" type="D" value="42156.9421574653"/>
        <Prp name="error count" type="I" value="0"/>
        <Prp name="warning count" type="I" value="0"/>
        <Prp name="root file name" type="S" value="RootLogData.dat"/>
        <Prp name="root logdata name" type="S" value="Orcamento"/>
        <Prp name="completed" type="B" value="-1"/>
        <Prp name="computer name" type="S" value="VM-WIN7"/>
        <Prp name="status" type="I" value="0"/>
        <Prp name="user name" type="S" value="Goku"/>
    </Node>
</Nodes>
```

Através desse padrão foi possível utilizar a pesquisa em arquivos por expressões regulares. Na Figura 19 demonstra-se um *print screen* da pesquisa por expressão regular para encontrar o *user name*, o qual indica quem é o responsável pela execução dos testes.

Figura 19 – Pesquisa por expressão regular

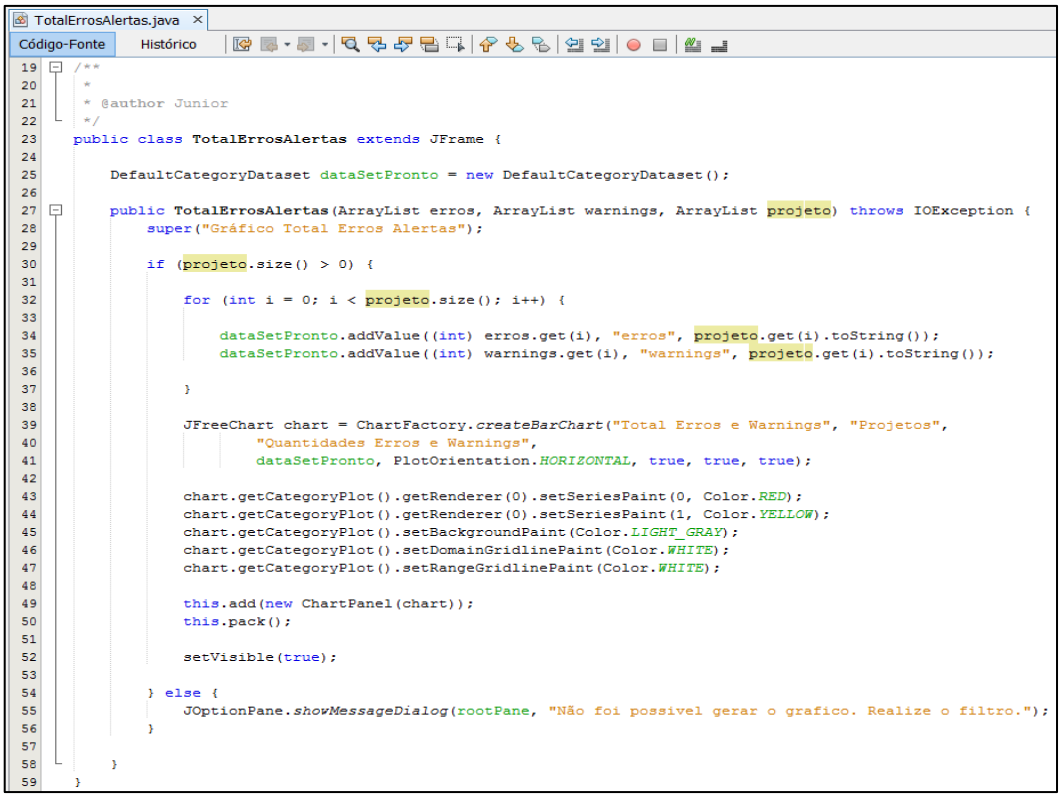


3.3.1.5 Framework JFreeChart

Framework JFreeChart é uma biblioteca livre para a plataforma Java, que pode ser utilizada para gerar gráficos de pizza, gráficos de barra, gráficos de linha, dentre diversos outros tipos de gráficos (JFREECHART, 2005). Estes podem ser exportados para o formato PNG ou JPEG.

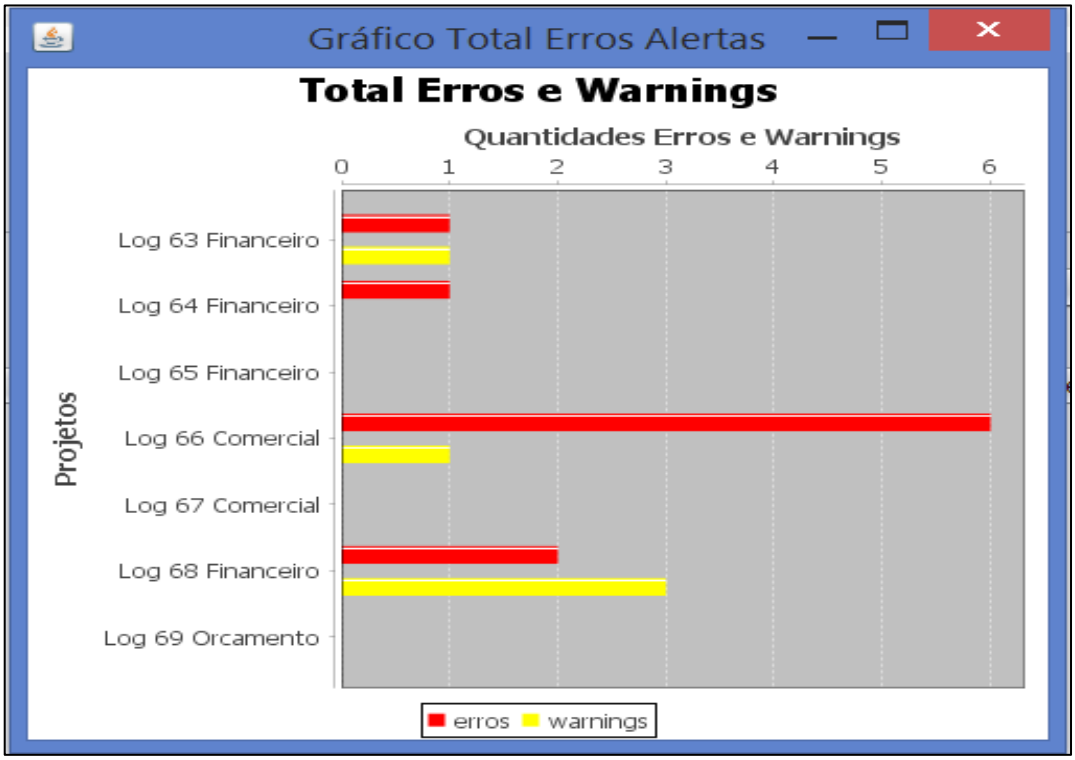
Na Figura 20 demonstra-se um *print screen* da classe Java `TotalErrosAlertas` que cria um gráfico de barra.

Figura 20 – Classe Java gráfico de Barra



Na Figura 21 demonstra-se um *print screen* do gráfico de barras gerado pela classe demonstrada na Figura 20.

Figura 21 – Gráfico de barras

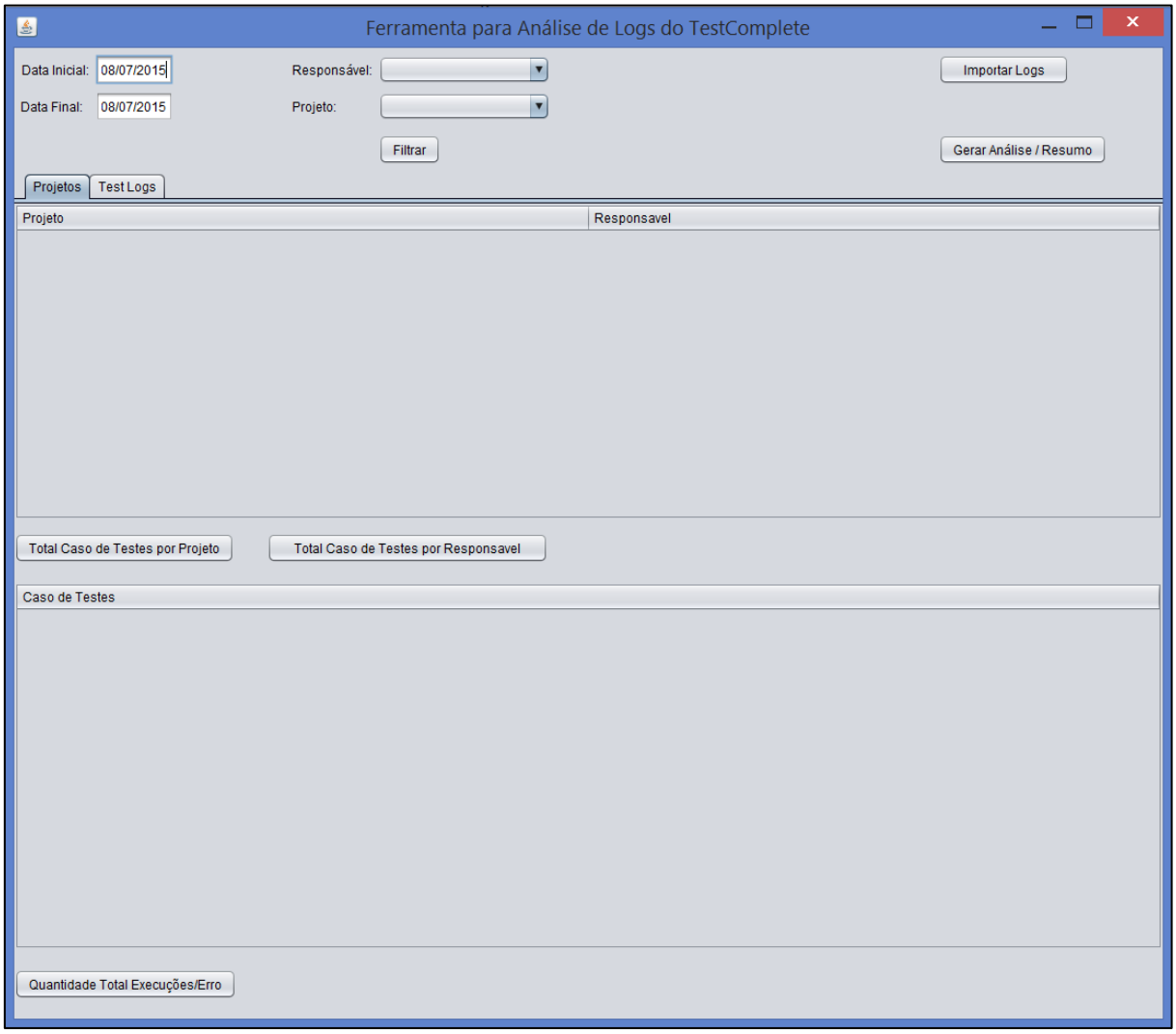


3.3.2 Operacionalidade da implementação

Nesta subseção é demonstrado o funcionamento da implementação através da utilização do sistema por meio do usuário.

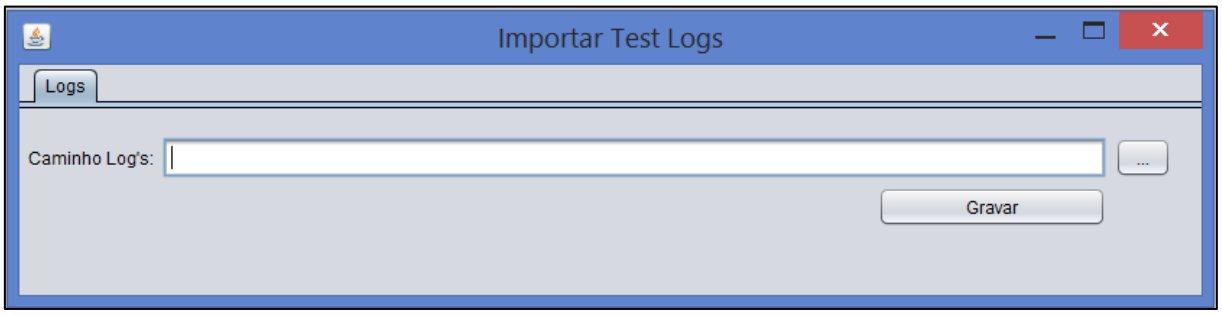
Inicialmente, ao abrir a ferramenta, é apresentada a sua tela principal, conforme mostra a Figura 22.

Figura 22 – Tela principal



Na tela principal, o usuário pode importar os *Test Logs* através do botão `Importar Logs`. Ao selecioná-lo é apresentada a tela Importar *Test Logs*, conforme mostra a Figura 23.

Figura 23 – Tela Importar *Test Logs*



A tela Importar *Test Logs* possui o campo `caminho Log's`, no qual deve-se informar o caminho onde se encontram os arquivos de *Test Logs* que se desejam importar.

Pode ser informada uma pasta que contenha vários arquivos de *Test Logs*, ou simplesmente indicar apenas um único arquivo de *Test Logs*. Este caminho pode ser digitado ou selecionado através do botão "`...`" ao lado do campo.

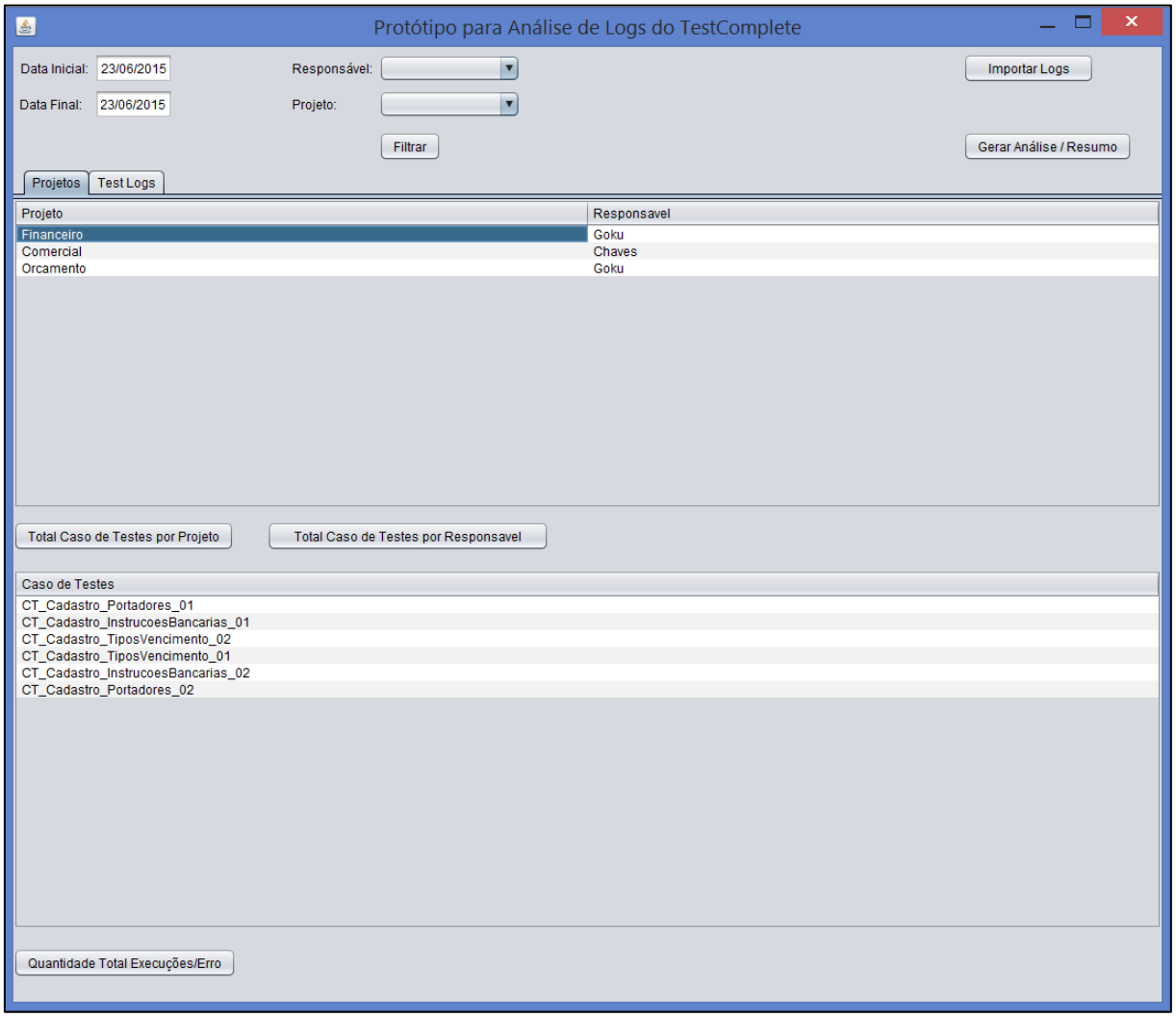
Ao selecionar o botão `Gravar`, a ferramenta realiza uma pesquisa para encontrar todos os arquivos de *Test Logs* no caminho informado. Antes de realizar a inserção no banco de dados é verificado se o(s) mesmo(s) *Test Logs* já não foi(ram) importado(s) anteriormente. O *Test Log* é considerado já importado quando já existe no banco de dados um *Test Log* do mesmo projeto com data e horários iguais aos que estão sendo importados.

Após realizar a importação é possível filtrar os *Test Logs* por uma data inicial e final em que foram executados, responsável pelo projeto e por projeto. Estes valores devem ser informados nos seguintes campos:

a) `data inicial`: este campo é preenchido automaticamente com a data atual, mas a ferramenta permite que o usuário modifique-a, pois este campo refere-se à data em que um *Test Log* foi executado;
b) `data final`: este campo é preenchido automaticamente com a data atual, mas a ferramenta permite que o usuário modifique-a, pois este campo se refere à data final em que um *Test Log* foi executado;
c) `responsável`: deve ser selecionado um responsável que é carregado automaticamente. Para visualizar os *Test Logs* de todos os responsáveis deve ser selecionada a opção em branco. Ao selecionar um usuário, a ferramenta automaticamente traz somente os projetos deste responsável no campo Projeto;
d) `projeto`: deve ser selecionado um projeto, que é carregado automaticamente. Para visualizar os *Test Logs* de todos os projetos deve ser selecionada a opção em branco.

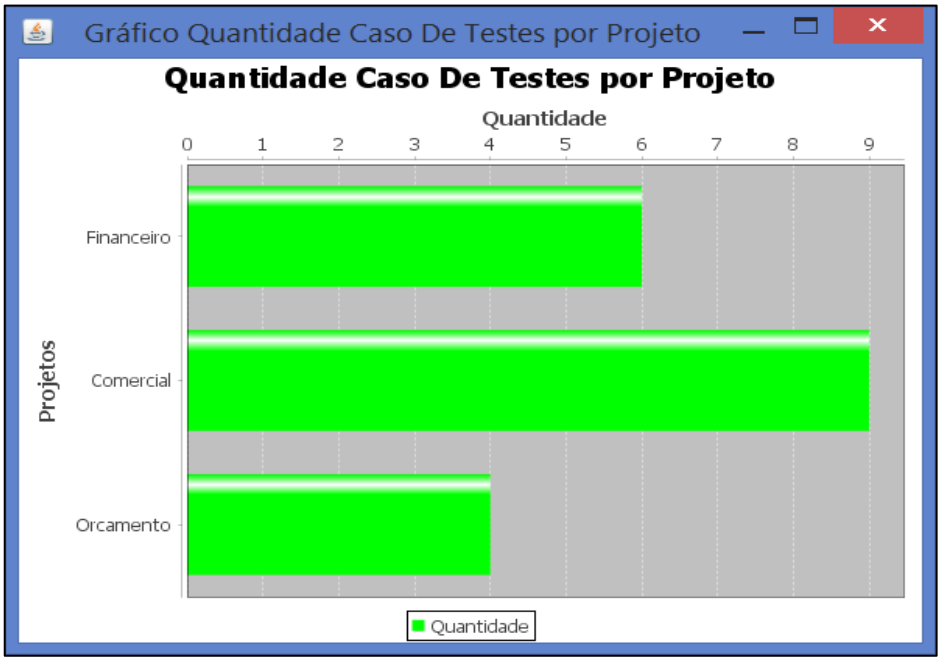
Ao selecionar o botão `Filtrar` da tela principal, a ferramenta busca todos os projetos e apresenta-os na *grid* da aba Projetos. Ao selecionar um projeto, são carregados automaticamente todos os casos de testes vinculados a esse projeto na *grid* Caso de Testes, conforme mostra a figura 24.

Figura 24 – Projetos e Caso de Testes filtrados



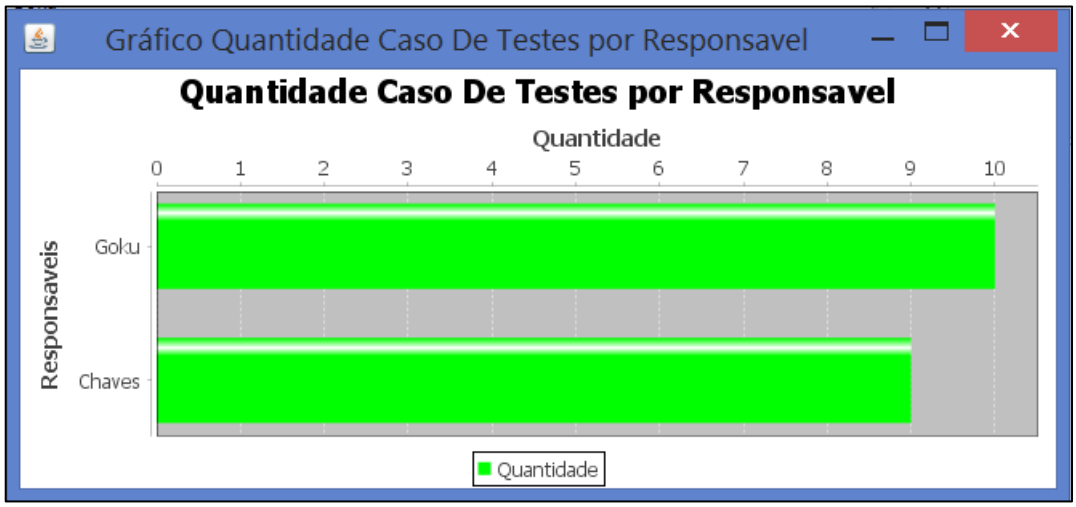
Com os dados listados nas *grids* é possível gerar os gráficos com informações referentes aos projetos e casos de testes. Ao selecionar o botão `Total Caso de Testes por Projeto` é apresentado o gráfico Quantidade Caso de Testes por Projeto, conforme mostra a Figura 25.

Figura 25 – Gráfico Quantidade Caso de Testes por Projeto



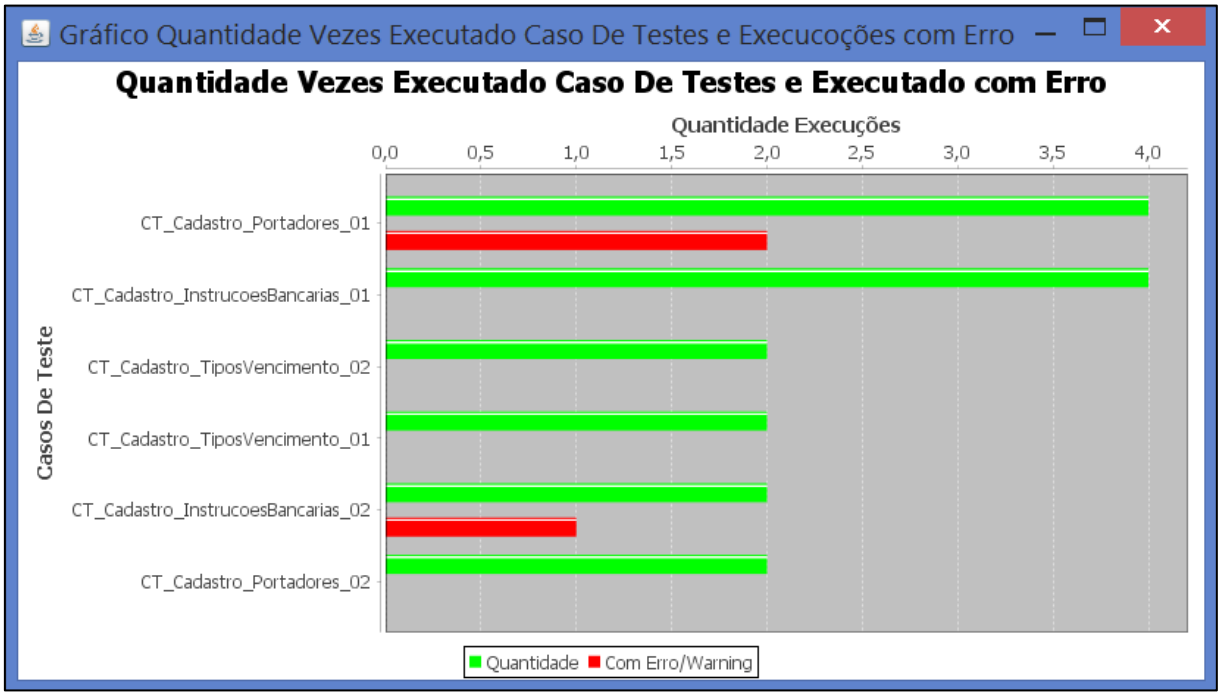
É possível também gerar um gráfico com a quantidade total de casos de testes por cada responsável da equipe de testes. Ao selecionar o botão `Total Caso de Testes por Responsável` é apresentado o gráfico, conforme mostra a Figura 26.

Figura 26 – Gráfico Quantidade Caso de Testes por Responsável



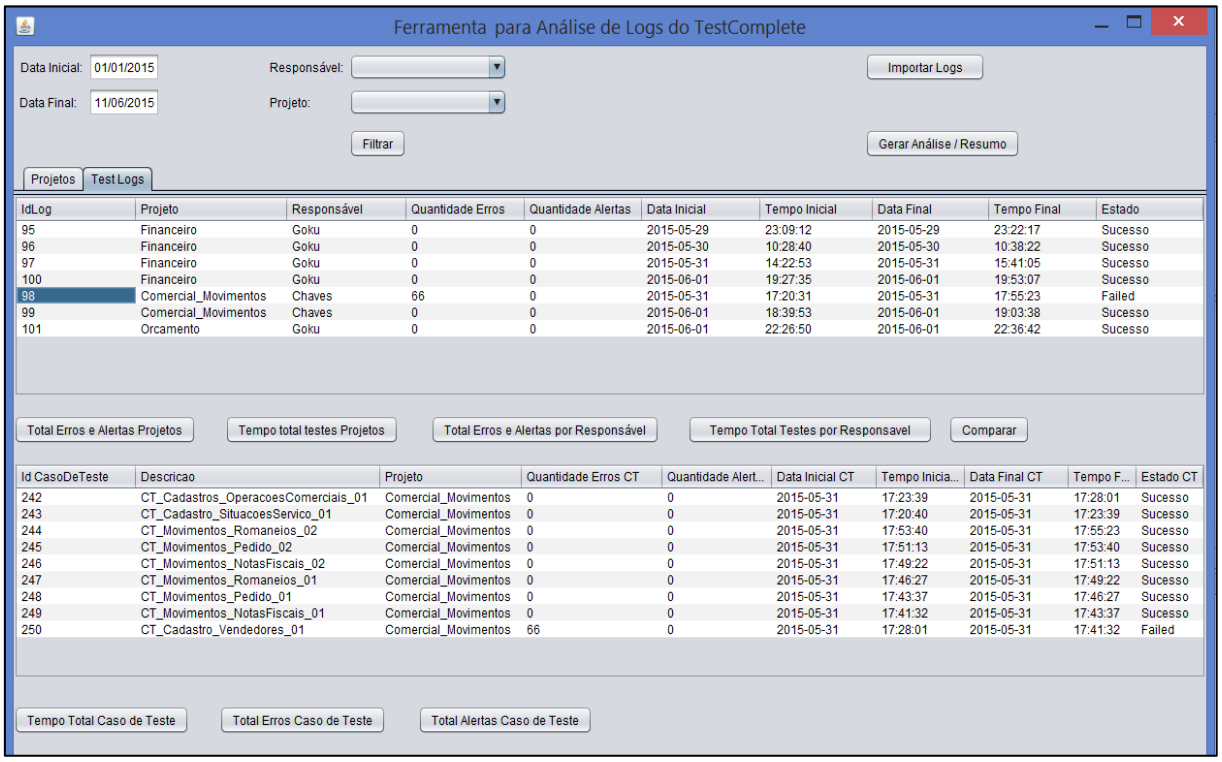
Selecionando um projeto é possível gerar o gráfico (Figura 27) que apresenta o total de execuções dos casos de testes relacionados a esse projeto, bem como também um total de execuções com erros, sendo assim possível identificar um caso de teste com histórico de problemas maiores que os demais. Este gráfico é apresentado após selecionar o botão `Quantidade Total Execuções/Erro`.

Figura 27 – Gráfico Total execuções caso de testes



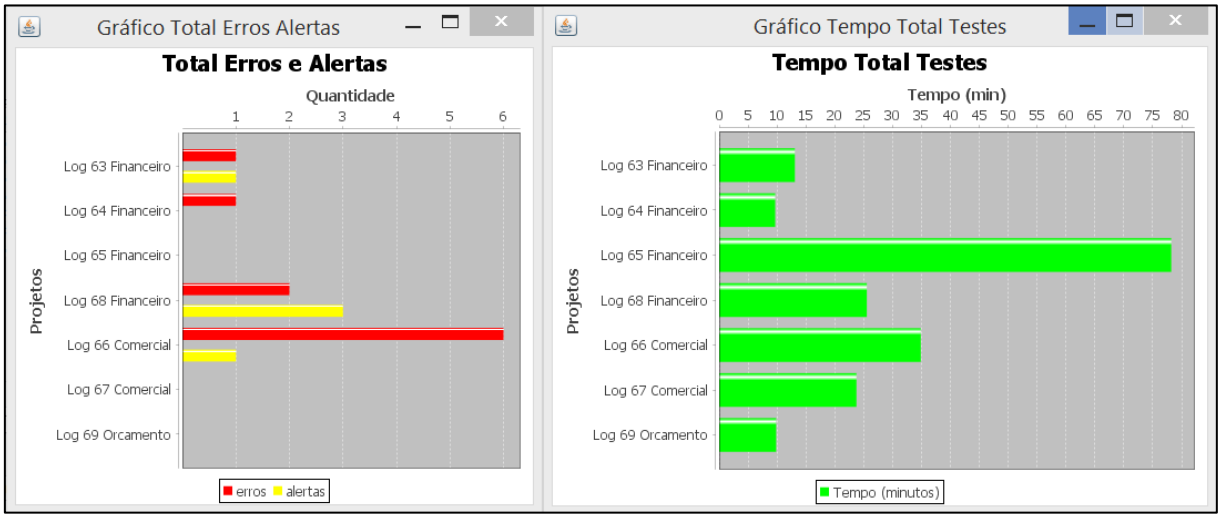
Na aba *Test Logs* são apresentados os logs e seus resultados de cada execução de acordo com os dados informados nos campos de filtro. Ao selecionar um *Test Log* na *grid Logs*, são automaticamente carregados os casos de testes executados na *grid* Resultado Caso de Testes, conforme Figura 28.

Figura 28 – *Test Logs* e resultado das execuções



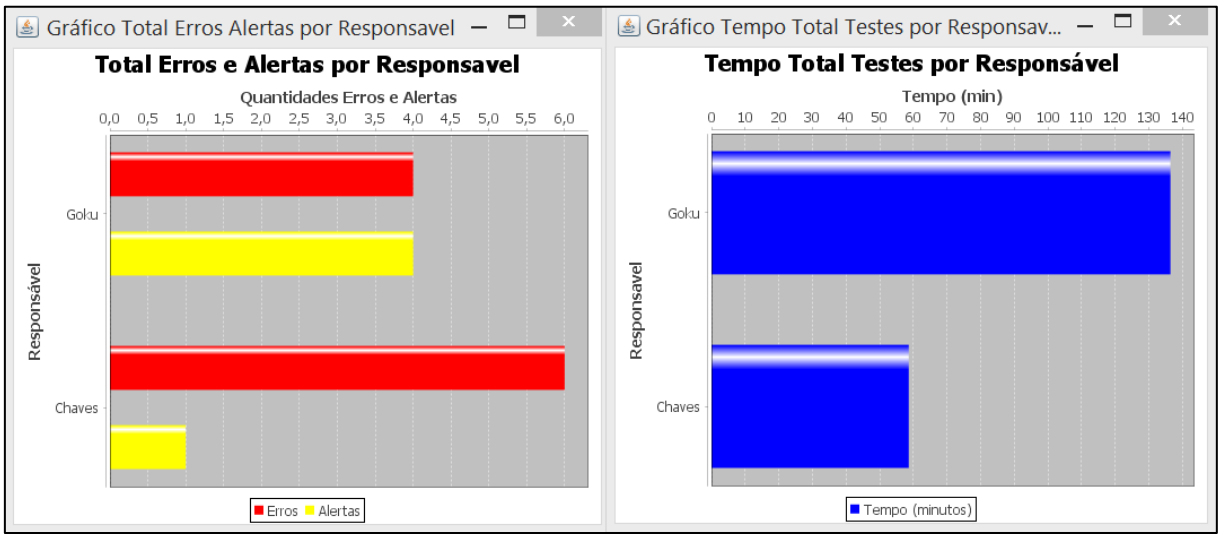
Com os *Test Logs* listados na *grid* é possível gerar os gráficos para melhor visualização dos resultados. Podem ser gerados gráficos das execuções por projetos ou por responsáveis. Na Figura 29 são apresentados os gráficos com o total de erros e alertas, e o gráfico de tempo total dos testes, ambos por projeto.

Figura 29 – Gráficos por projetos dos resultados dos *Test Logs*



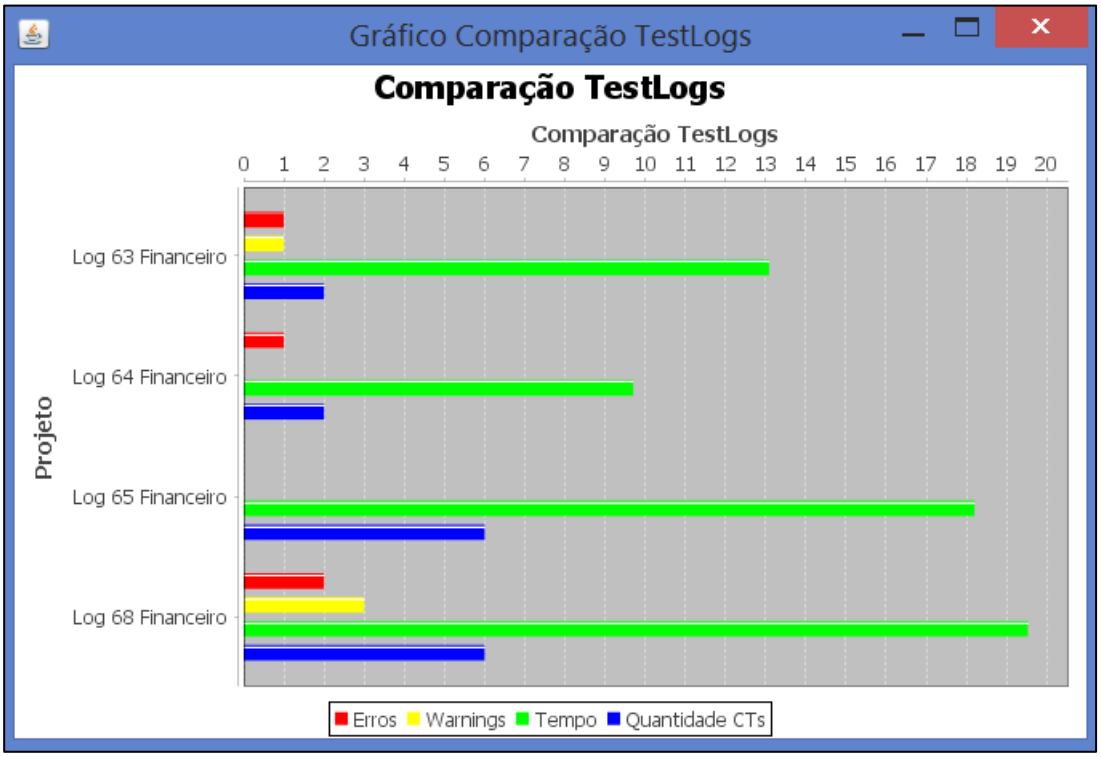
Na Figura 30 são apresentados os gráficos com o total de erros e alertas, e o gráfico de tempo total dos testes, ambos por responsáveis.

Figura 30 – Gráficos por responsáveis dos resultados dos *Test Logs*



É possível comparar dois ou mais *Test Logs*. Para realizar essa comparação é necessário filtrar os *Test Logs* apenas de um único projeto. Na Figura 31 é apresentado um exemplo do gráfico com a comparação de vários *Test Logs* do mesmo projeto.

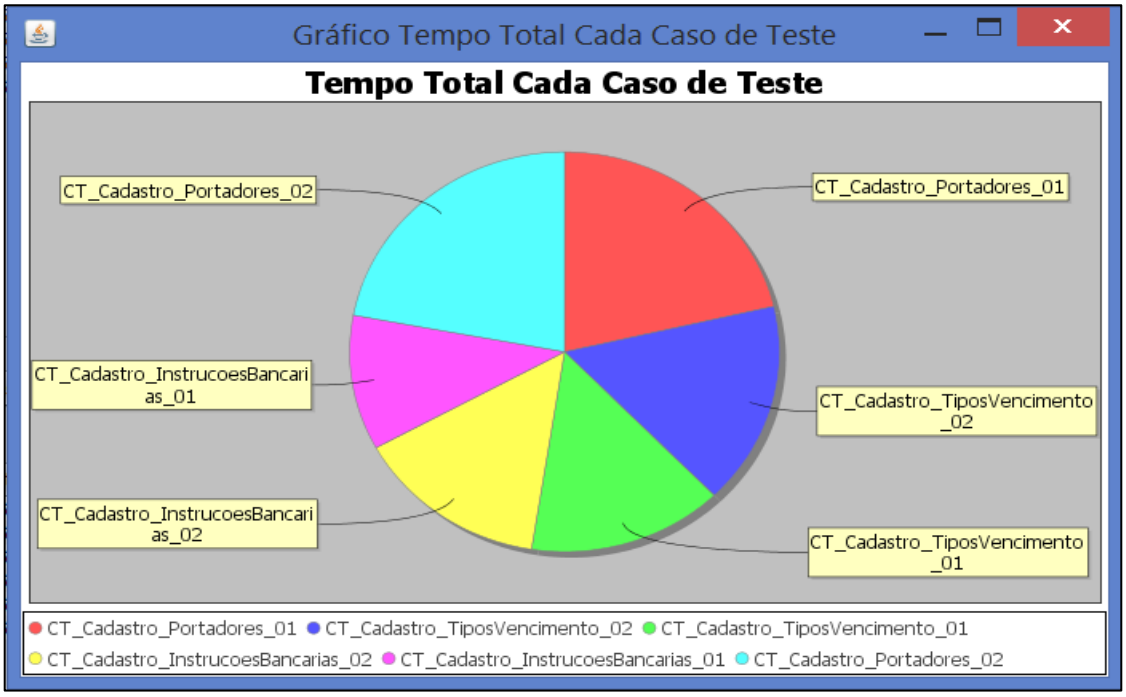
Figura 31 – Gráfico de comparação de vários *Test Logs*



Para a análise dos casos de testes executados em cada *Test Logs*, é realizada a geração de gráficos em formato pizza. Só é possível gerar os gráficos após selecionar um *Test Logs*.

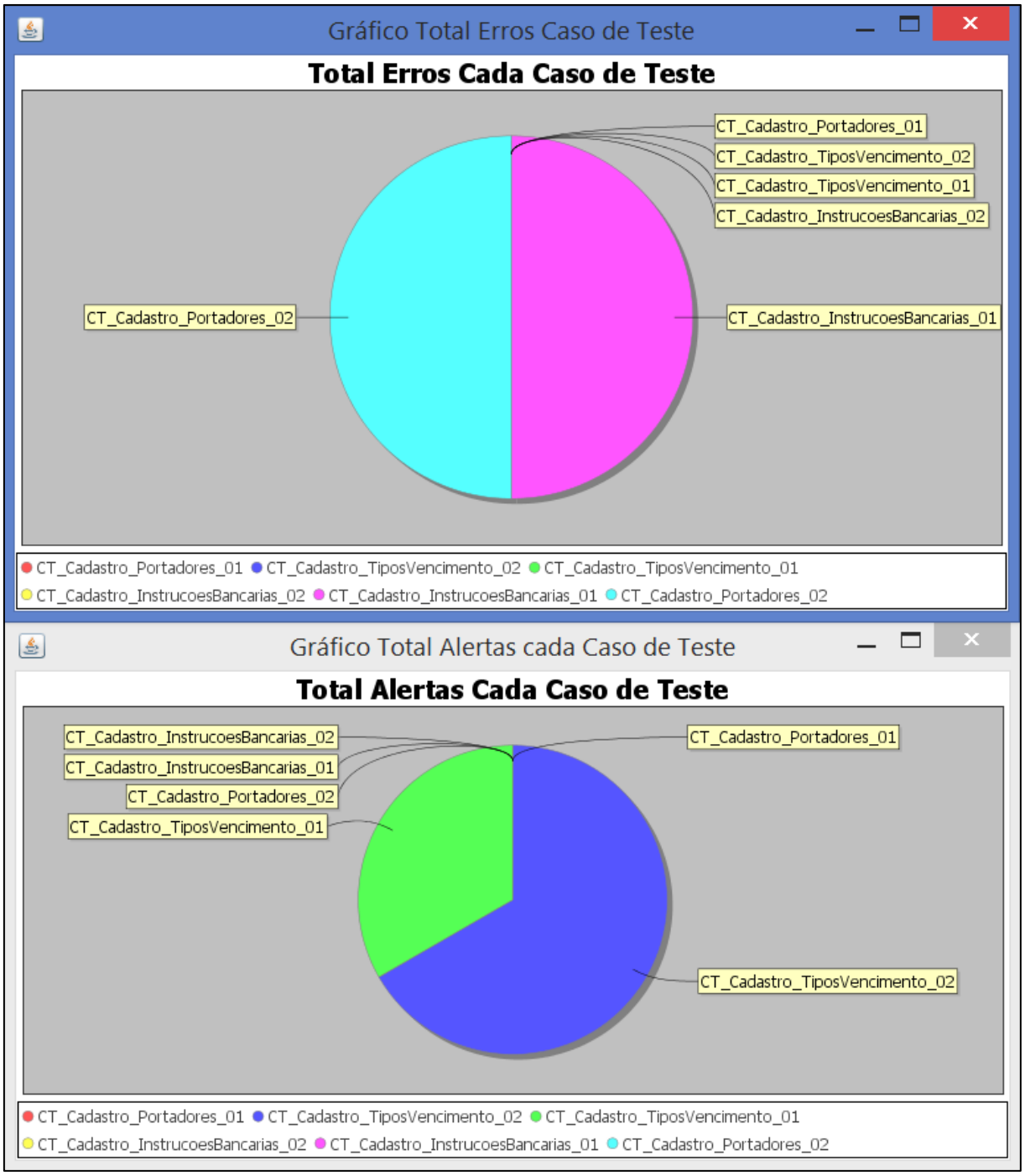
Ao selecionar o botão `Tempo Total Caso de Teste` é apresentado o gráfico com o tempo total de cada caso de teste executado, sendo possível analisar com maior facilidade o caso de teste com maior tempo de execução, conforme Figura 32.

Figura 32 – Gráfico tempo total cada caso de teste



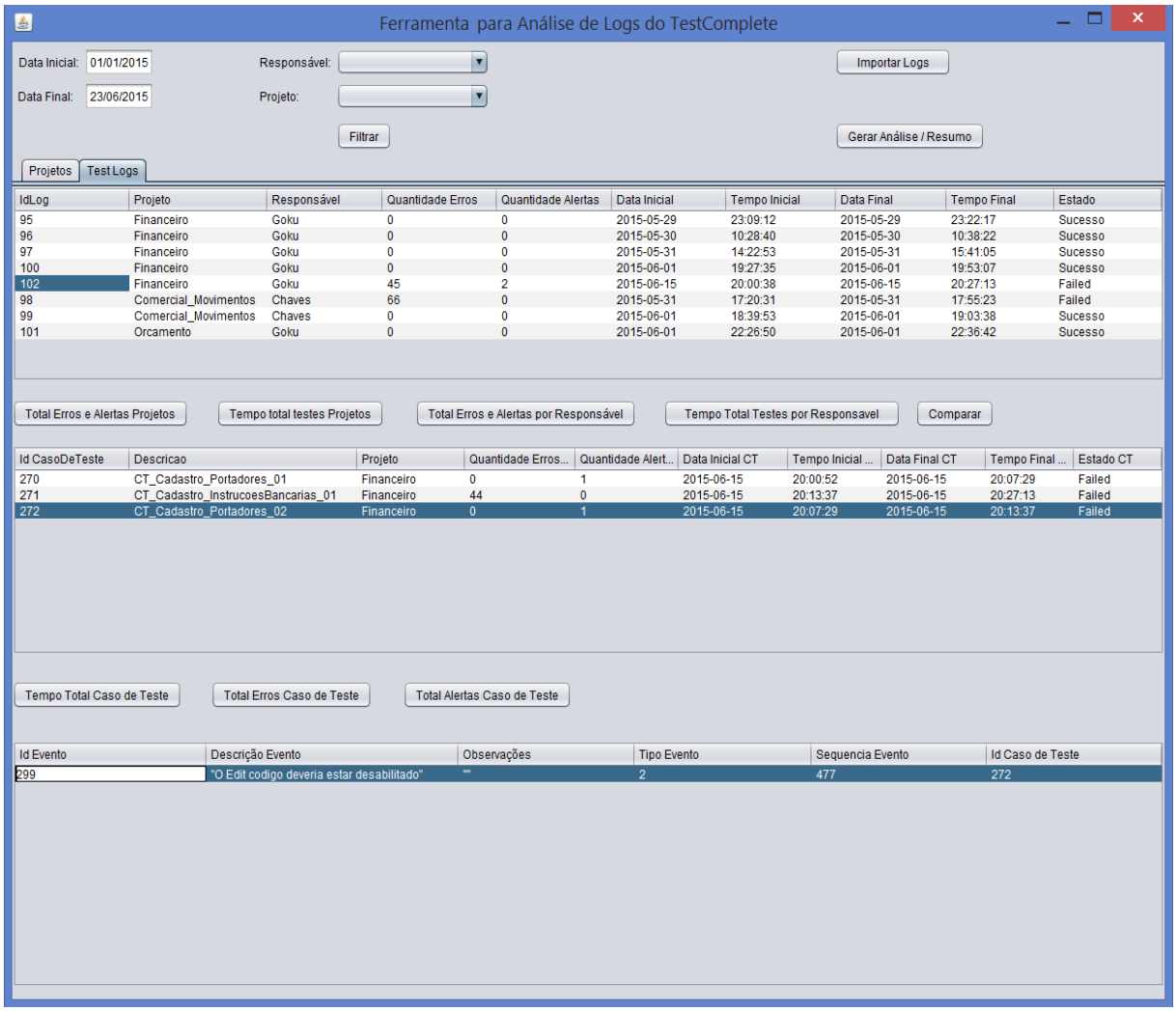
É possível visualizar o total de erros e alertas de cada caso de teste, selecionando os botões `Total Erros Caso de Teste` e `Total Alertas Caso de Teste`. A ferramenta irá gerar os gráficos em formato pizza, conforme apresentado na Figura 33.

Figura 33 – Gráfico Total Erros e Alertas cada Caso de Teste



Quando um caso de teste executado apresentar erros ou alertas, é possível visualizar os eventos que causaram o erro ou alerta. Ao selecionar o caso de teste com algum problema é listado na *grid* Evento um detalhamento dessa situação, conforme apresentado na Figura 34.

Figura 34 – Eventos caso de testes com erros ou alertas



A ferramenta também é capaz de gerar um arquivo PDF com as imagens dos principais gráficos. Para gerá-lo é necessário ter informações listadas na *grid* Projeto da aba Projetos e *Test Logs* na aba *Test Logs*. Ao selecionar o botão `Gerar Análise / Resumo` a ferramenta apresenta uma tela para informar o caminho onde deseja salvar o arquivo PDF. As imagens dos gráficos gerados no arquivo PDF são apresentadas no Anexo A. Ao selecionar o botão `OK` o arquivo PDF é gerado com os seguintes gráficos:

a) quantidade Caso de Testes por Projeto;
b) quantidade Caso de Testes por Responsável;
c) total Erros e Alertas por projeto;
d) tempo Total Testes por projeto;
e) total Erros e Alertas por responsáveis;
f) tempo total Testes por responsáveis.

3.4 RESULTADOS E DISCUSSÃO

O Quadro 3 apresenta a comparação entre a Ferramenta para Análise de *Logs* do *TestComplete* com os trabalhos correlatos, apresentados na seção 2.5.

Quadro 3 – Quadro comparativo do sistema com os trabalhos correlatos

Características	**Ferramenta desenvolvida**	**MARCOS (2007)**	**WALTRICK (2011)**
Linguagem de programação	Java	Delphi	Pascal / Delphi Script
Banco de dados	MySQL	Nenhum	MySQL
Ambiente (desktop/web)	Desktop	Desktop	Desktop
Ferramenta para o *TestComplete*	SIM	SIM	SIM
Automatiza um processo	SIM, visualização de vários *Test Logs*	SIM, analisa telas e gera *script* de testes.	SIM, integra resultados das execuções no *TestLink*

Os trabalhos correlatos e o desenvolvimento neste trabalho não possuem a mesma finalidade, mas eles têm como objetivo automatizar um processo para auxiliar os usuários da ferramenta TestComplete. O trabalho de Marcos (2007) analisa as telas geradas no Delphi e cria os scripts de testes que servem como o passo inicial para criação dos casos de testes específicos para cada tela. Já o trabalho de Waltrick (2011) foi criar um *plugin* que automatiza o processo de registrar o resultado da execução de um teste na ferramenta *TestLink*. Ambos utilizaram a ferramenta *TestComplete*, e tiveram sua parcela de contribuição com relação ao uso da ferramenta propriamente dita.

Convém destacar as diferenças entre os trabalhos apresentados e a ferramenta desenvolvida quanto ao uso de tecnologias. O trabalho de Marcos (2007) foi desenvolvido utilizando Delphi e nenhum banco de dados, já Waltrick (2011) utilizou Pascal e Delphi com banco de dados MySQL.

Com relação aos objetivos específicos deste trabalho, todos foram considerados atingidos, haja vista que a partir da utilização da ferramenta ao gerar os gráficos Quantidade Caso de Testes por Responsável e Quantidade Total Execuções, o gestor da qualidade consegue ter um acompanhamento geral da equipe de testes em relação à quantidade de casos de testes e execuções de cada testador. Foram realizados testes informais e verificado que a ferramenta além de auxiliar a análise e comparação dos *Test Logs* reduziu consideravelmente o tempo gasto para realizar esta atividade. Com a ferramenta *TestComplete* o tempo médio para se analisar o resultado de dez projetos é de 30 minutos, com a utilização da ferramenta criada o tempo médio para realizar esta mesma análise passou a ser de 5 minutos.

4 CONCLUSÕES

Neste trabalho apresentou-se o desenvolvimento de uma ferramenta na plataforma *desktop* para auxiliar os usuários da ferramenta *TestComplete* a analisar e comparar os resultados dos *Test Logs*, cujos objetivos iniciais foram alcançados com sucesso.

As ferramentas utilizadas para a elaboração do trabalho mostraram-se adequadas. A utilização da biblioteca *JFreeChart* contribuiu com sua facilidade de gerar vários tipos de gráficos e por desenhar automaticamente a escala de eixos e legendas. O maior desafio na produção do trabalho foi na leitura dos arquivos *TestLogs*. Para tal, foi necessário criar um padrão genérico utilizando expressões regulares para identificar e buscar somente as informações úteis para a análise e comparação dos *Test Logs*.

Ao desenvolver a ferramenta foi considerado o fato de que o mesmo deveria facilitar e agilizar as atividades de análise e comparação dos *Test Logs*, para isso foram desenvolvidas interfaces simples e padronizadas.

A versão do *TestComplete* é um limitador. A ferramenta está preparado para importar os arquivos de *TestLogs* gerados pela versão 7, pois foi a única versão que foi possível obter uma licença de uso *free*. Para novas versões, possivelmente, a ferramenta necessitará de ajustes.

Pode-se dizer que a conclusão deste trabalho atingiu os objetivos pessoais, pois propiciaram ao autor a aplicação de boa parte dos conhecimentos adquiridos ao longo do curso de graduação.

4.1 EXTENSÕES

A partir deste projeto, como sugestão para trabalhos futuros tem-se:

a) implementar uma versão *web*, para permitir ao gestor da qualidade acessar a ferramenta de onde estiver;
b) desenvolver um *plugin* para adicionar o *TestLog* selecionado na ferramenta *TestComplete* e abrí-lo para verificar algumas informações adicionais de cada execução;
c) desenvolver um filtro para informar hora inicial e final da execução, pois no projeto

foi considerado que haveria apenas uma execução por dia de cada projeto.

REFERÊNCIAS

BARTIÉ, A. **Garantia da qualidade de software.** Brasil, Rio de Janeiro: Campus, 2002.

BASTOS, A; RIOS, E; CRISTALLI, R; MOREIRA, T. **Base de conhecimento em teste de software.** Brasil, São Paulo: Martins, 2007.

CAETANO, C. **Automação e gerenciamento de testes: aumentando a produtividade com as principais soluções open source e gratuitas.** Florianópolis, 2007. Disponível em: <http://www.linhadecodigo.com.br/artigo/1566/automacao-e-gerenciamento-de-testes-aumentando-a-produtividade-com-as-principais-solucoes-open-source-e-gratuitas-2a-edicao.aspx>. Acesso em: 23 fev. 2015.

DEVMEDIA. **Iniciando Expressões Regulares.** [S.l], 2007. Disponível em: < http://www.devmedia.com.br/iniciando-expressoes-regulares/6557>. Acesso em: 23 fev. 2015.

DUSTIN, E. **Automated Software Testing.** [S.l], [2015?]. Disponível em: < http://www.asq509.org/ht/a/GetDocumentAction/i/50550>. Acesso em: 26 fev. 2015.

GRAHAM, D.; FEWSTER, M. **Software test automation**. Inglaterra, Londres: Addison-Wesley, 1999.

HAYES, L. G. **Evolution of automated software testing.** Maryland, 2009. Disponível em: <http://www.automatedtestinginstitute.com/home/ASTMagazine/2009/AutomatedSoftwareTestingMagazine_August2009.pdf>. Acesso em: 23 fev. 2015.

JFREECHART. **Welcome To JFreeChart.** [S.l], 2005. Disponível em: < http://www.jfree.org/jfreechart/>. Acesso em: 23 fev. 2015.

MARCOS, Adriana F. **Ferramenta de apoio à automatização de testes através do testcomplete para programas desenvolvidos em delphi.** 2007. 76 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MOLINARI, L. **Inovação e automação de testes de software.** Brasil, São Paulo: Érica, 2010.

MOTA, A. **O que é TestComplete.** Nova Friburgo, 2011. Disponível em: <http://software123testando.wordpress.com/2011/06/06/o-que-e-o-testcomplete/ >. Acesso em: 23 fev. 2015.

RICE, R. W. **Laying a foundation for automation.** Maryland, 2009. Disponível em: <http://www.automatedtestinginstitute.com/home/ASTMagazine/2009/AutomatedSoftwareTestingMagazine_May2009.pdf>. Acesso em: 23 fev. 2015.

SMARTBEAR. **Teste Automatizado – TestComplete.** Beverly, 2013a. Disponível em: <http://smartbear.com/products/qa-tools/automated-testing-tools>. Acesso em: 23 fev. 2015.

_____. **Automated Test Reporting – TestComplete.** Beverly, 2013b. Disponível em: <http://smartbear.com/products/qa-tools/automated-testing-tools/software-test-run-reporting>. Acesso em: 23 fev. 2015.

WALTRICK, Douglas O. **Plugin da ferramenta TestComplete para integração com a ferramenta testlink.** 2011. 61 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

APÊNDICE A – DESCRIÇÃO DOS CASOS DE USO

Este Apêndice apresenta a descrição dos principais casos de uso descritos na seção de especificação deste trabalho. Nos Quadros de 4 até 17 são demonstrados todos os casos de uso definidos para o desenvolvimento do sistema.

Quadro 4 – Descrição do caso de uso importar *Test Logs*

Caso de uso UC01 – Importar *Test Logs*.
Ator: Gestor da qualidade/Testador.
Objetivo: Importar *Test Logs*.
Pré-condições: Nenhuma.
Pós-condições: O Gestor da Qualidade/Testador importou um ou vários *Test Logs*.

Cenário Principal:
1. O gestor da qualidade/testador seleciona o botão Importar *Logs*;
2. A ferramenta apresenta a tela Importar Test Logs;
3. O gestor da qualidade/testador informa o caminho dos *Test Logs* no campo Caminho Log's;
4. O gestor da qualidade/testador seleciona o botão gravar;
5. A ferramenta grava as informações dos *Test Logs* encontrados.

Cenário Exceção – Se no passo 3 do cenário principal o caminho informado não conter Test Logs:
- A ferramenta exibe uma mensagem de aviso indicando que no caminho informado não existem Test Logs para serem importados.

Quadro 5 – Descrição do caso de uso gerar análise

Caso de uso UC02 – gerar análise.
Ator: Gestor da qualidade/Testador.
Objetivo: gerar análise dos Test Logs em um arquivo PDF.
Pré-condições: realizar o filtro dos *Test Logs* na tela principal.
Pós-condições: O gestor da qualidade/testador gerou a análise em um arquivo PDF.

Cenário Principal:
1. O gestor da qualidade/testador seleciona o botão Gerar Análise / Resumo;
2. A ferramenta apresenta a tela de Gerar Análise / Resumo Test Logs;
3. O gestor da qualidade/testador informa o caminho do PDF a ser gerado no campo Caminho Arquivo;
4. O gestor da qualidade/testador seleciona o botão Ok;
5. A ferramenta gera o PDF no caminho informado no passo 3.

Quadro 6 – Descrição do caso de uso visualizar quantidade caso de testes por projeto

Caso de uso UC03 – Visualizar quantidade de caso de testes por projeto.
Ator: Gestor da qualidade/Testador.

Objetivo: Visualizar um gráfico em barras com a quantidade de caso de testes em cada projeto.
Pré-condições: realizar o filtro dos Test Logs na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:
1. O gestor da qualidade/testador seleciona o botão Total Caso de Testes por Projeto;
2. A ferramenta gera o gráfico Quantidade Caso de Testes por Projeto.

Cenário Exceção – Se no passo 1 do cenário principal não tiver projetos carregados na *grid*:
- A ferramenta exibe uma mensagem de aviso indicando que não foi possível gerar o gráfico.

Quadro 7 – Descrição do caso de uso visualizar quantidade caso de testes por responsável

Caso de uso UC04 – Visualizar quantidade de caso de testes por responsável.
Ator: Gestor da qualidade/Testador.
Objetivo: Visualizar um gráfico em barras com a quantidade de caso de testes por responsável.
Pré-condições: realizar o filtro dos Test Logs na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:
1. O gestor da qualidade/testador seleciona o botão Total Caso de Testes por Responsável;
2. A ferramenta gera o gráfico Quantidade Caso de Testes por Responsável.

Cenário Exceção – Se no passo 1 do cenário principal não tiver projetos carregados na *grid*:
- A ferramenta exibe uma mensagem de aviso indicando que não foi possível gerar o gráfico.

Quadro 8 – Descrição do caso de uso visualizar totais execuções caso de testes

Caso de uso UC05 – Visualizar totais execuções caso de testes.
Ator: Gestor da qualidade/Testador.
Objetivo: Visualizar um gráfico em barras com o total de execuções de cada caso de teste de um determinado projeto.
Pré-condições: realizar o filtro dos Test Logs na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:
1. O gestor da qualidade/testador seleciona um projeto na grid Projetos.
2. A ferramenta carrega a grid Caso de Testes com todos os casos de testes do projeto selecionado.
3. O gestor da qualidade/testador seleciona o botão Quantidade Total Execuções/Erro;
4. A ferramenta gera o gráfico Quantidade vezes executado Caso de Testes e executados com erro.

Cenário Exceção – Se no passo 3 do cenário principal não tiver casos de testes carregados na *grid*:

- A ferramenta exibe uma mensagem de aviso indicando que não foi possível gerar o gráfico.

Quadro 9 – Descrição do caso de uso visualizar totais execuções caso de testes com erro

Caso de uso UC06 – Visualizar totais execuções caso de testes com erro.
Ator: Gestor da qualidade/Testador.
Objetivo: Visualizar um gráfico em barras com o total de execuções de cada caso de teste com erro de um determinado projeto.
Pré-condições: realizar o filtro dos Test Logs na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:
1. O gestor da qualidade/testador seleciona um projeto na grid Projetos.
2. A ferramenta carrega a grid Caso de Testes com todos os casos de testes do projeto selecionado.
3. O gestor da qualidade/testador seleciona o botão Quantidade Total Execuções/Erro;
4. A ferramenta gera o gráfico Quantidade vezes executado Caso de Testes e executado com erro.

Cenário Exceção – Se no passo 3 do cenário principal não tiver *Test Logs* carregados na *grid*:
- A ferramenta exibe uma mensagem de aviso indicando que não foi possível gerar o gráfico.

Quadro 10 – Descrição do caso de uso visualizar quantidade erros e alertas dos projetos

Caso de uso UC07 – Visualizar quantidade de erros e alertas projetos.
Ator: Gestor da qualidade/Testador.
Objetivo: Visualizar um gráfico em barras com a quantidade de erros e alertas dos projetos.
Pré-condições: realizar o filtro dos Test Logs na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:
1. O gestor da qualidade/testador seleciona o botão Total Erros e Alertas Projetos;
2. A ferramenta gera o gráfico Total Erros e Alertas.

Cenário Exceção – Se no passo 1 do cenário principal não tiver *Test Logs* carregados na *grid*:
- A ferramenta exibe uma mensagem de aviso indicando que não foi possível gerar o gráfico.

Quadro 11 – Descrição do caso de uso visualizar tempo total de testes por projeto

Caso de uso UC08 – Visualizar tempo total de testes por projeto.
Ator: Gestor da qualidade/Testador.
Objetivo: Visualizar um gráfico em barras com o tempo total de testes por projeto.
Pré-condições: realizar o filtro dos Test Logs na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:

1. O gestor da qualidade/testador seleciona o botão Tempo total testes Projetos;
2. A ferramenta gera o gráfico Tempo Total Testes.

Cenário Exceção – Se no passo 1 do cenário principal não tiver *Test Logs* carregados na *grid*:
- A ferramenta exibe uma mensagem de aviso indicando que não foi possível gerar o gráfico.

Quadro 12 – Descrição do caso de uso visualizar quantidade erros e alertas por responsável

Caso de uso UC09 – Visualizar quantidade de erros e alertas por responsável.
Ator: Gestor da qualidade/Testador.
Objetivo: Visualizar um gráfico em barras com a quantidade de erros e alertas por responsável.
Pré-condições: realizar o filtro dos Test Logs na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:
1. O gestor da qualidade/testador seleciona o botão Total Erros e Alertas por Responsável;
2. A ferramenta gera o gráfico Total Erros e Alertas por Responsável.

Cenário Exceção – Se no passo 1 do cenário principal não tiver *Test Logs* carregados na *grid*:
- A ferramenta exibe uma mensagem de aviso indicando que não foi possível gerar o gráfico.

Quadro 13 – Descrição do caso de uso visualizar tempo total testes por responsável

Caso de uso UC10 – Visualizar tempo total de testes por responsável.
Ator: Gestor da qualidade/Testador.
Objetivo: Visualizar um gráfico em barras com o tempo total de testes por responsável.
Pré-condições: realizar o filtro dos Test Logs na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:
1. O gestor da qualidade/testador seleciona o botão Tempo total testes por Responsável;
2. A ferramenta gera o gráfico Tempo Total Testes por Responsável.

Cenário Exceção – Se no passo 1 do cenário principal não tiver *Test Logs* carregados na *grid*:
- A ferramenta exibe uma mensagem de aviso indicando que não foi possível gerar o gráfico.

Quadro 14 – Descrição do caso de uso comparar dois ou mais *Test Logs*

Caso de uso UC11 – Comparar dois ou mais *Test Logs*.
Ator: Gestor da qualidade/Testador.
Objetivo: Comparar dois ou mais *Test Logs* do mesmo projeto.
Pré-condições: realizar o filtro dos Test Logs do mesmo projeto na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:

1. O gestor da qualidade/testador seleciona o botão Comparar;
2. A ferramenta gera o gráfico Comparação *Test Logs*.

Cenário Exceção – Se no passo 1 do cenário principal os Test Logs filtrados não pertencem ao mesmo projeto:

- A ferramenta exibe uma mensagem de aviso indicando que os *Test Logs* não podem ser comparados, pois não são do mesmo projeto filtrado.

Quadro 15 – Descrição do caso de uso visualizar tempo total cada caso de testes

Caso de uso UC12 – Visualizar tempo total cada caso de testes.
Ator: Gestor da qualidade/Testador.
Objetivo: Visualizar um gráfico de formato pizza com o tempo total de cada caso de testes.
Pré-condições: realizar o filtro dos Test Logs na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:

1. O gestor da qualidade/testador seleciona um Test Log na *grid* Test Logs.
2. A ferramenta carrega a grid Resultado Execução Caso de Testes com todos os resultados dos casos de testes do Test Log selecionado.
3. O gestor da qualidade/testador seleciona o botão Tempo Total cada Caso de Teste;
4. A ferramenta gera o gráfico Tempo Total Caso de Teste.

Cenário Exceção – Se no passo 3 do cenário principal não tiver Casos de testes executados carregados na *grid*:

- A ferramenta exibe uma mensagem de aviso indicando que não foi possível gerar o gráfico.

Quadro 16 – Descrição do caso de uso visualizar total de erros cada caso de teste

Caso de uso UC13 – Visualizar total de erros cada caso de testes.
Ator: Gestor da qualidade/Testador.
Objetivo: Visualizar um gráfico de formato pizza com o total de erros de cada caso de testes.
Pré-condições: realizar o filtro dos Test Logs na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:

1. O gestor da qualidade/testador seleciona um Test Log na *grid* Test Logs.
2. A ferramenta carrega a grid Resultado Execução Caso de Testes com todos os resultados dos casos de testes do Test Log selecionado.
3. O gestor da qualidade/testador seleciona o botão Total Erros Caso de Teste;
4. A ferramenta gera o gráfico Total Erros cada Caso de Teste.

Cenário Exceção – Se no passo 3 do cenário principal não tiver Casos de testes executados carregados na *grid*:

- A ferramenta exibe uma mensagem de aviso indicando que não foi possível gerar o gráfico.

Quadro 17 – Descrição do caso de uso visualizar total de alertas cada caso de teste

Caso de uso UC14 – Visualizar total de alertas cada caso de testes.
Ator: Gestor da qualidade/Testador.
Objetivo: Visualizar um gráfico de formato pizza com o total de alertas de cada caso de testes.
Pré-condições: realizar o filtro dos *Test Logs* na tela principal.
Pós-condições: O gestor da qualidade/testador visualiza o gráfico com as informações.

Cenário Principal:
1. O gestor da qualidade/testador seleciona um *Test Log* na *grid* Test Logs.
2. A ferramenta carrega a grid Resultado Execução Caso de Testes com todos os resultados dos casos de testes do Test Log selecionado.
3. O gestor da qualidade/testador seleciona o botão Total Alertas Caso de Teste;
4. A ferramenta gera o gráfico Total Alertas cada Caso de Teste.

Cenário Exceção – Se no passo 3 do cenário principal não tiver Casos de testes executados carregados na *grid*:
- A ferramenta exibe uma mensagem de aviso indicando que não foi possível gerar o gráfico.

APÊNDICE B – DESCRIÇÃO DO DICIONÁRIO DE DADOS

Este Apêndice apresenta a descrição das tabelas do banco de dados apresentadas na seção de especificação deste trabalho. Os tipos de dados utilizados nos atributos são:

a) *integer*: armazena numéricos inteiros de 32 *bits*;

b) *varchar*: armazena caracteres alfanuméricos até 255 caracteres;

c) *dateTime*: armazena caracteres alfanuméricos com formato de data e hora;

Nos Quadros de 18 até 23 é demonstrado o dicionário de dados das tabelas.

Quadro 18 – Dicionário da tabela "responsável"

Entidade: Responsavel		
Descrição: Armazena responsáveis da ferramenta		
Atributo	**Tipo**	**Descrição**
idResponsavel	*integer*	Chave primária
nome	*varchar(45)*	Nome Responsável

Quadro 19 – Dicionário da tabela "projeto"

Entidade: Projeto		
Descrição: Armazena projetos da ferramenta		
Atributo	**Tipo**	**Descrição**
idProjeto	*integer*	Chave primária
descricao	*varchar(45)*	Descrição projeto
Responsavel_idReponsavel	*integer*	Chave estrangeira, entidade responsavel

Quadro 20 – Dicionário da tabela "casoDeTeste"

Entidade: CasoDeTeste		
Descrição: Armazena caso de testes da ferramenta		
Atributo	**Tipo**	**Descrição**
idDescricaoCasoDeTeste	*integer*	Chave primária
descricao	*varchar(100)*	Descrição Caso de Teste
Projeto_IdProjeto	*Integer*	Chave estrangeira, entidade projeto

Quadro 21 – Dicionário da tabela “log”

Entidade: Log		
Descrição: Armazena Test Logs da ferramenta		
Atributo	**Tipo**	**Descrição**
idLogs	*integer*	Chave primária
errorCount	*integer*	Quantidade erros
warningCount	*integer*	Quantidade alertas
startTime	*dateTime*	Data e hora do inicio da execução
stopTime	*dateTime*	Data e hora do término da execução
status	*varchar(45)*	Status da execução
Projeto_IdProjeto	*Integer*	Chave estrangeira, entidade projeto

Quadro 22 – Dicionário da tabela “logCasoDeTeste”

Entidade: LogCasoDeTeste		
Descrição: Armazena resultados das execuções dos casos de testes		
Atributo	**Tipo**	**Descrição**
idCasoDeTeste	*integer*	Chave primária
errorCountCT	*integer*	Quantidade erros no caso de teste
warningCountCT	*integer*	Quantidade alertas no caso de teste
startTimeCT	*dateTime*	Data e hora do inicio da execução do caso de teste
stopTimeCT	*dateTime*	Data e hora do término da execução do caso de teste
statusCT	*varchar(45)*	Status da execução
idDescricaoCasoDeTeste	*integer*	Chave estrangeira, entidade casoDeTeste
log_idLogs	*integer*	Chave estrangeira, entidade log

Quadro 23 – Dicionário da tabela “eventocasodeteste”

Entidade: EventoCasodeTeste		
Descrição: Armazena os eventos dos caso de testes executados com falhas.		
Atributo	**Tipo**	**Descrição**
ideventoscasodeteste	*integer*	Chave primária
descricaoEvento	*varchar(1000)*	Descrição do evento ocorrido
descricaoRemarks	*Varchar(1000)*	Descrição do *remarks* gerado no momento do erro ou alerta
typeEventos	*integer*	Tipo do Evento
eventoId	*integer*	Id gerado no *Test Log* que permite a ordenação dos eventos
Logcasodeteste_idlogct	*integer*	Chave estrangeira, entidade LogCasoDeTeste

ANEXO A – ARQUIVO PDF GERADO NA ANÁLISE/RESUMO

Neste anexo apresenta-se o arquivo PDF com as imagens dos principais gráficos da ferramenta. Nas Figuras de 35 até 40 são apresentadas as imagens do PDF gerado com os gráficos.

Figura 35 – PDF gerado com o gráfico Quantidade Caso de Testes por Projeto

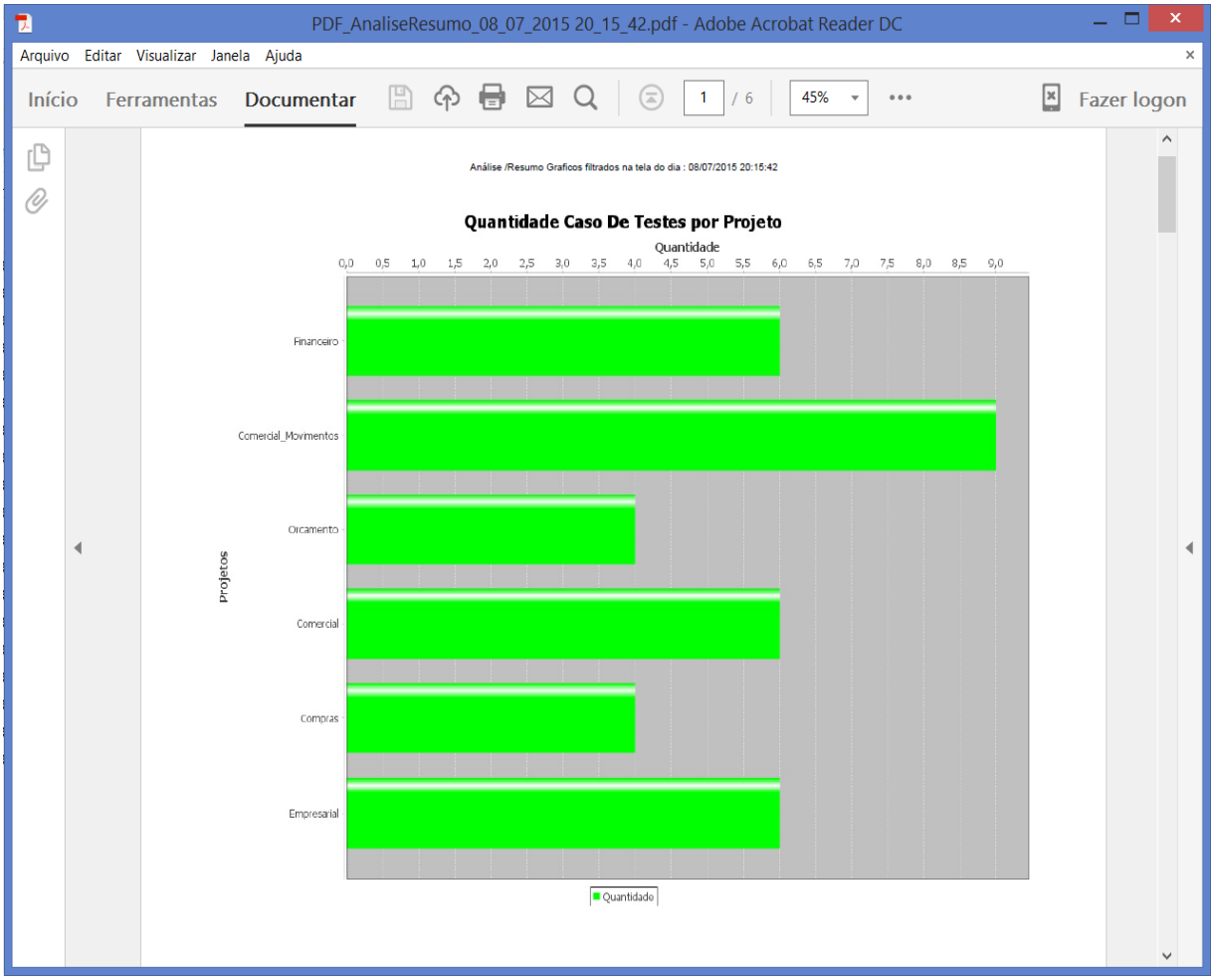


Figura 36 – PDF gerado com o gráfico Quantidade Caso de Testes por Responsável

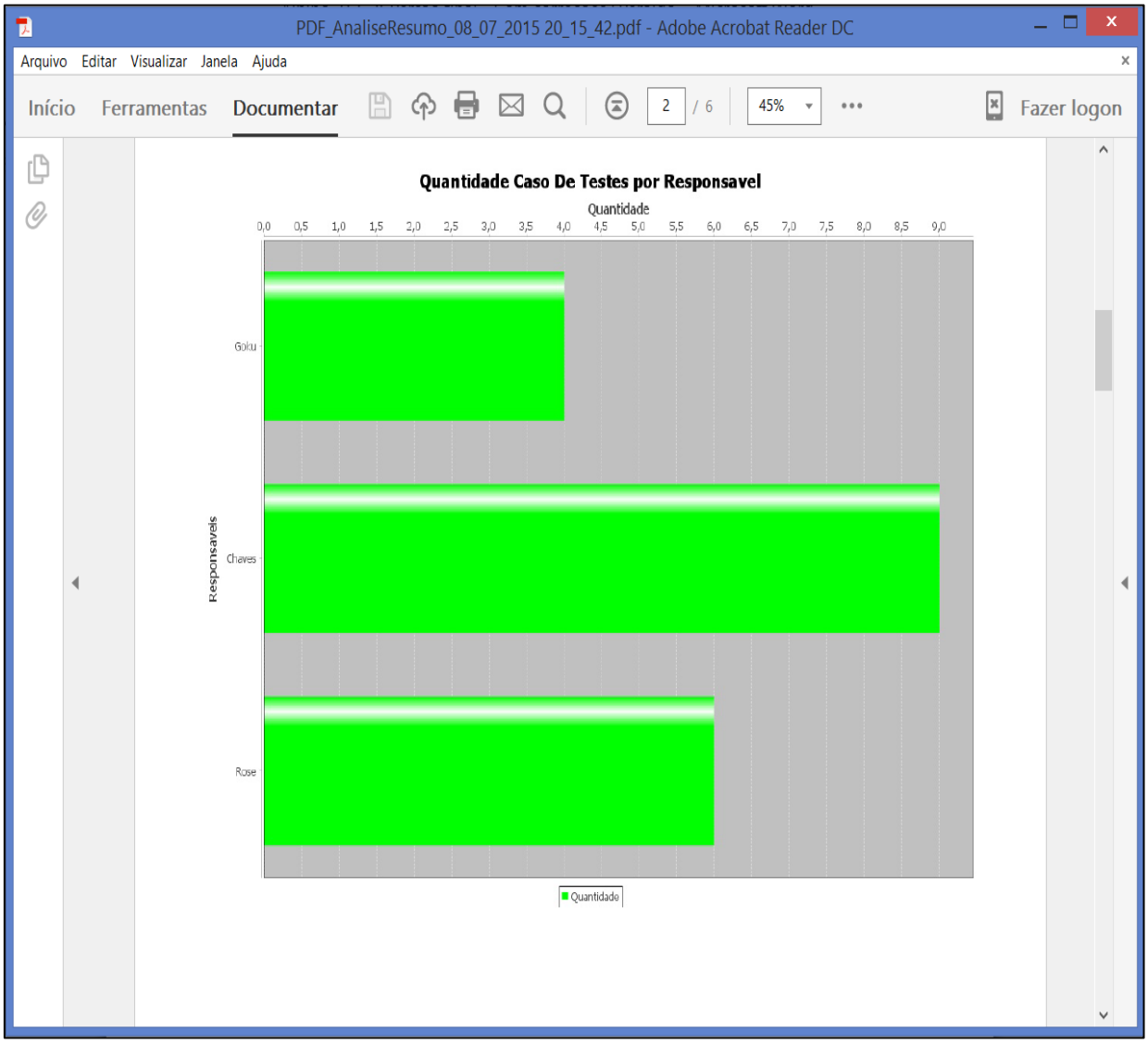


Figura 37 – PDF gerado com o gráfico Total Erros e Alertas por Projeto

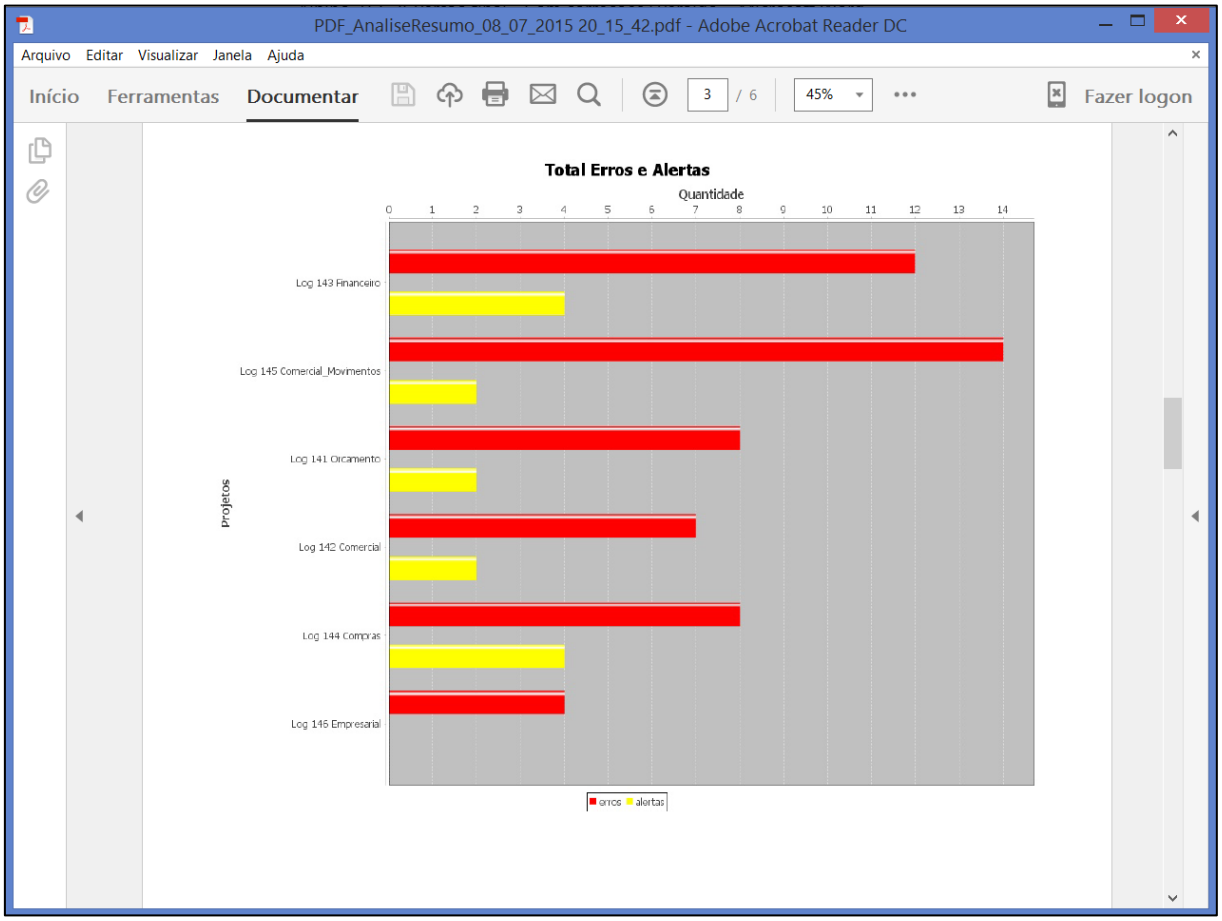


Figura 38 – PDF gerado com o gráfico Tempo Total Testes por Projeto

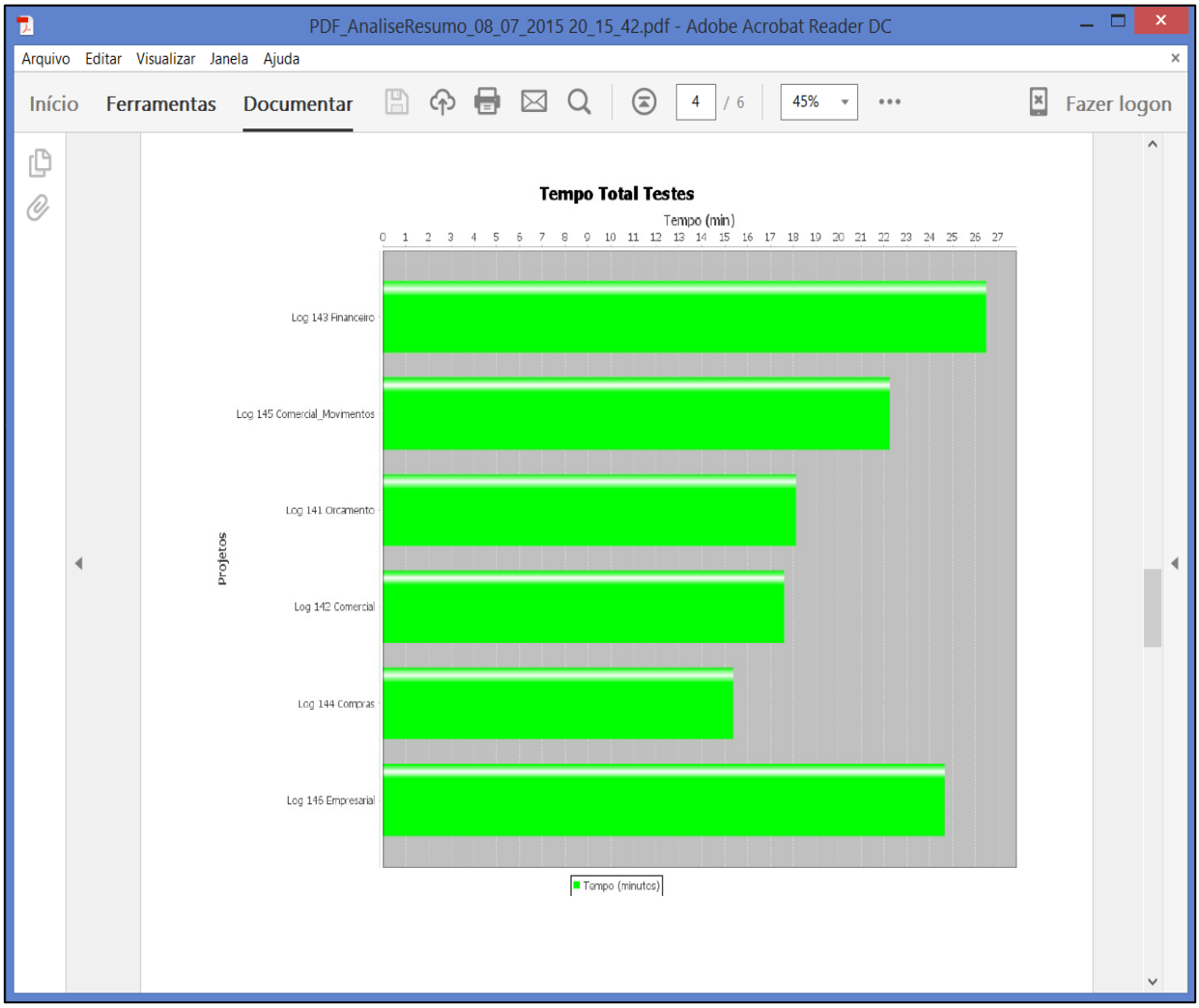


Figura 39 – PDF gerado com o gráfico Total Erros e Alertas por Responsáveis

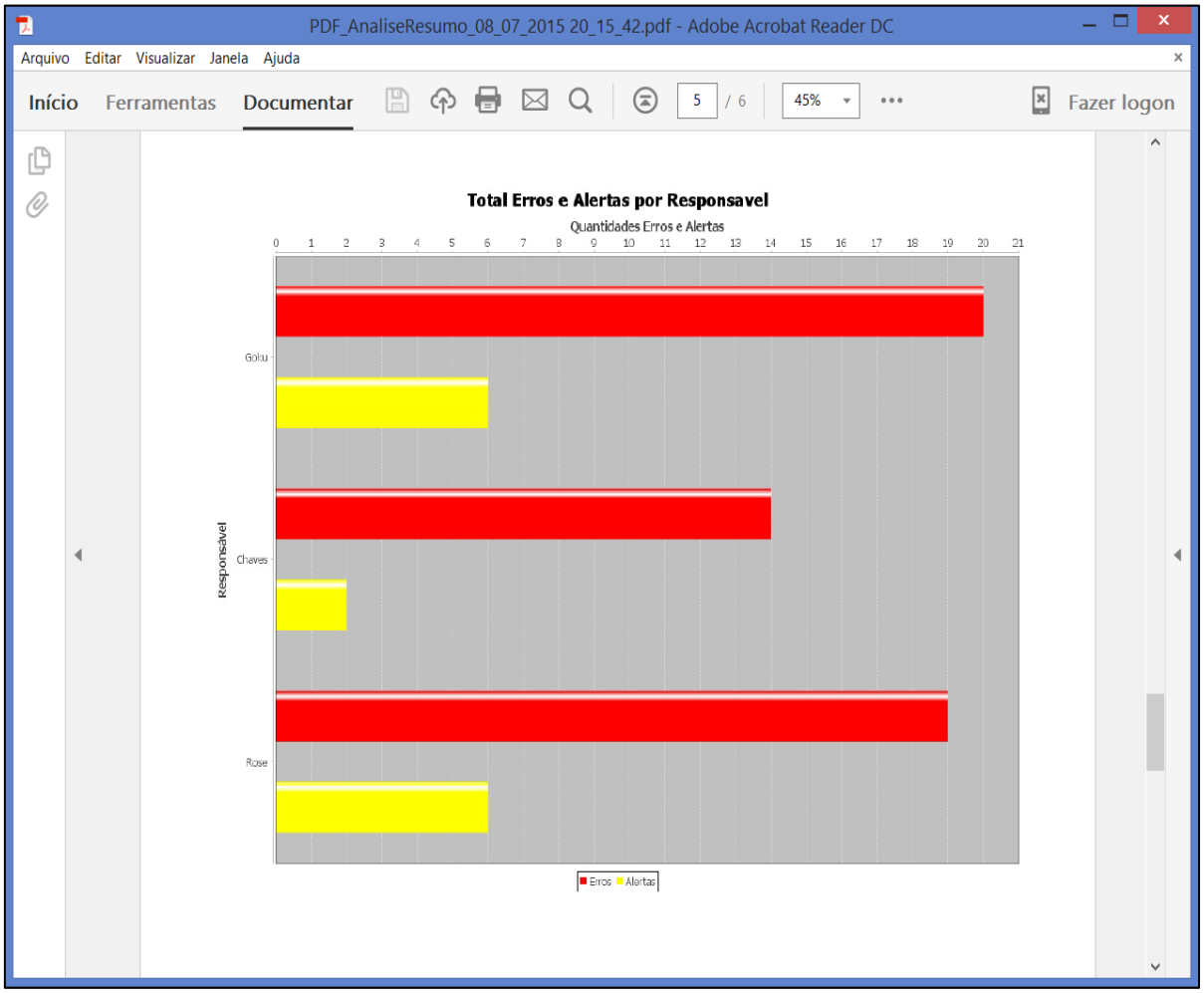


Figura 40 – PDF gerado com o gráfico Tempo Total Testes por Responsáveis