Coordenação do Curso de Sistemas de Informação
Universidade Estadual de Mato Grosso do Sul

FERRAMENTA PARA GERAÇÃO DE CASO DE TESTE FUNCIONAL

José Roberto do Amaral

Profa. Msc. Jéssica Bassani de Oliveira(Orientadora)

Novembro de 2017

FERRAMENTA PARA GERAÇÃO DE CASO DE TESTE FUNCIONAL

José Roberto do Amaral

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso II devidamente corrigida e defendida por José Roberto do Amaral e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Sistemas de Informação.

Dourados, 6 de novembro de 2017.

Profa. Msc. Jéssica Bassani de Oliveira
(Orientadora)

Coordenação do Curso de Sistemas de Informação
Universidade Estadual de Mato Grosso do Sul

FERRAMENTA PARA GERAÇÃO DE CASO DE TESTE FUNCIONAL

José Roberto do Amaral

Novembro de 2017

Banca Examinadora:

- Profa. Msc. Jéssica Bassani de Oliveira (Orientadora)
  Área de Computação - UEMS
- Profa. Dra. Gláucia Gabriel Sass
  Área de Computação - UEMS
- Prof. Dr. Cleber Valgas Gomes Mira
  Área de Computação - UEMS

Agradecimentos

Primeiramente a Deus, que guiou-me em mais essa etapa da minha vida.
A minha esposa Jacimara pela paciência e compreensão. Sempre esteve ao meu lado com seu apoio incondicional, me aconselhando para que eu pudesse conseguir meus objetivos. O meu muito obrigado.
A minha orientadora, professora Msc. Jéssica Bassani, a esta valiosa oportunidade, pela orientação, dedicação e conhecimentos repassados. Os meus sinceros agradecimentos.

Resumo

Fazer os documentos para caso de teste é complexo. Uma ferramenta que suporte a criação de casos de teste pode contribuir para a eficiência do projeto de *software*. Este estudo possui o objetivo de criar uma ferramenta para geração de caso de teste funcional que consiste em determinar um roteiro de teste que será aplicado para suprir a deficiência e análise do teste. Desenvolver esta ferramenta requer conhecimento em técnicas de teste e linguagem de programação. A ferramenta tem suporte exclusivamente para teste em *Desktop*. Os resultados com esta ferramenta são: maior produtividade na elaboração dos documentos de testes, credibilidade e êxito na execução das tarefas. Esses resultados foram obtidos através da simulação da ferramenta através de um estudo de caso.

Palavras-chave: *Teste de Software*, *Caso de Teste*, *Teste Funcional*, *C#*

Abstract

Making the documents for test cases is complex. A tool that supports the creation of test cases can contribute to software design efficiency. This work has the objective of creating a tool for a functional test case that consists of determining a test roadmap that will be applied to deficiency and test analysis. Developing this tool requires knowledge in test techniques and programming language. The tool has support for testing on Desktop only. The results with this tool are: productivity in the preparation of test documents, credibility and success in the execution of tasks. These results were obtained through the simulation of the tool through a case study.

Key-words: *Software Test*, *Test Case*, *Functional Test*, *C#*

Sumário

Lista de Siglas

IEEE *Instituto de Engenheiros Eletricistas e Eletrônicos*

IDE *Ambiente de Desenvolvimento Integrado*

HTML *Linguagem de Marcação de Hipertexto*

HTTP *Protocolo de Transferência de Hipertexto*

API *Interface de Programação de Aplicativos*

DER *Diagrama Entidade-Relacionamento*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Durante o processo de desenvolvimento de *software* pode surgir algumas ou várias anomalias e que muitas vezes estão relacionadas as falhas, erros, performance e segurança, comprometendo o projeto e seus prazos. Compreender as técnicas e os processos de teste é penetrar em uma área do *software* pouco explorada e que, para os programadores parece mais como um campo minado.

Para resolver esta problemática, existem várias ferramentas e empresas especializadas que trabalham com os mais diversos tipos de teste de *software*. Por exemplo, a *Selenium* (http://www.seleniumhq.org), segundo o próprio site, é uma ferramenta que se aplica em testar componentes da *web*. Também existe o *Badboy* (http://www.badboy.com.au), que consiste em outra ferramenta gratuita para automatização de teste. No entanto, em ambos os casos os testes são feitos em aplicações *web*.

Estas ferramentas normalmente são complexas e de difícil entendimento, já que o ciclo do teste é tratado como um todo, incluindo em sua estrutura a maioria dos testes (unidade, estrutura, funcional, aceitação, etc). Em muitos casos, os usuários deverão informar para a ferramenta o que deve ser testado.

Para realizar o teste é preciso fazer a elaboração do que será testado, ou seja, o plano de teste. O plano de teste é um referencial com concepções sobre como deverão serem conduzido os testes. O plano é constituído por casos de teste que são documentos formados por um ou mais requisitos tendo em sua estrutura um maior detalhe do que vai ser testado (MOLINARI, 2008).

A elaboração manual do plano de teste pode ser onerosa para uma equipe de testadores de um projeto. Este projeto propõe o desenvolvimento de uma ferramenta que auxiliará na geração automática de Caso de Teste funcionais apresentando o comportamento do requisito a ser testado. A geração automática pode agilizar o processo de teste e diminuir falhas humanas durante o processo. De acordo com RIOS (2010), para desenvolver um sistema, um conjunto adequado de regras deve ser definido. Essa ideia também se aplica para os casos de teste.

1.1 Objetivo Geral

O objetivo geral deste estudo é construir uma ferramenta que será distribuída gratuitamente para criação de um Plano de Teste composto por Caso de Teste para aplicação em formulários de sistemas, dando um maior enfoque no teste funcional. A ferramenta não executará os casos de teste

1.1.1 Objetivos Específicos

- Determinar um roteiro de teste.
- Gerar relatórios com os possíveis casos de teste funcionais.
- Espera-se diminuir erros humanos no processo.
- Criar uma base de conhecimento para ser aplicada nos testes.
- Desenvolver Caso de Teste para as técnicas de Análise do Valor Limite, Tabela Verdade e Particionamento por Equivalência.

A ferramenta focou no teste funcional, o qual tem como qualidade preocupar-se apenas com os resultados esperados. Assim o testador poderá verificar se o resultado apresentado era o previsto. Caso validadas as informações, pode-se afirmar que determinada função cumpriu com o que foi proposto.

Diante da dificuldade em se testar um *software* e das complexidades em elaborar uma documentação de caso de teste funcional, a ferramenta tornará o trabalho produtivo e organizado, apresentando confiança na execução e consequentemente um maior êxito nas tarefas

1.2 Organização do trabalho

O trabalho é composto por 7 capítulos. O capítulo 1 apresenta a introdução e objetivos do trabalho. O capítulo 2 apresenta o referencial teórico utilizado neste projeto. No capítulo 3 apresentamos os trabalhos correlatos. Já o capítulo 4 apresenta metodologia da ferramenta construída. No capitulo 5 é exposto a implementação da ferramenta bem como screenshots das telas a fim de melhor demonstrar as interfaces. No capítulo 6 são apresentados os resultados. E, finalmente no capitulo 7 é discorrida a conclusão do projeto.

Capítulo 2

Fundamentação Teórica

Neste capítulo será apresentada uma breve descrição das tecnologias estudadas para a construção da ferramenta de criação de teste de *software*.

2.1 Teste de Software

A maioria dos produtos comercializados no Brasil, como por exemplo, eletrodomésticos, produtos de limpeza, materiais elétricos entre outros, antes de chegarem ao consumidor passam por rigorosos processos de testes para certificar que os mesmos estão de acordo com as conformidades exigidas pelo órgão competente, e dessa forma receber um selo de qualidade (INMETRO).

Com *software* não é diferente. Antes de ser colocado em produção, ou seja, antes de chegar ao usuário final, são feitos testes para sanar quaisquer defeitos ou erros que venham a prejudicar o seu funcionamento. Utiliza-se um conjunto de rotinas para averiguar certas anomalias no *software*, baseando-se em normas e padrões IEEE (Instituto de Engenheiros Elétricos e Eletrônicos). A mais essencial é o IEEE 829 (MOLINARI, 2008).

Por volta dos anos 70, a execução dos testes tinha como objetivo demonstrar que o produto funcionava e os testes eram feitos pelos desenvolvedores. Nos anos 80/90 os testes passaram a detectar e analisar se os requisitos foram atendidos, programadores e usuários eram quem executavam esses testes. Já nos anos 90/00 executava-se os testes para fins de prevenção, além das características acima citadas, também tinha o objetivo de garantir que o *software* não tinha defeito. Nesta época já eram utilizados processos de testes, e quem fazia os testes eram os desenvolvedores, usuários e testadores (RIOS, 2010).

De acordo MECENAS and OLIVEIRA (2005), o teste era executado com base em percepção e experiência de pessoas que projetam o sistema, não tendo nenhuma forma coordenada para que pudesse examinar o teste com mais eficácia.

Quando os programadores e analistas escrevem um programa, espera-se que o mesmo tenha um funcionamento apropriado de acordo com que foi descrito. Lamentavelmente, não é isso que acontece. Os sistemas são formados por inúmeras variantes, cálculos e algoritmos

que abrangem em sua maior parte muitos elementos e estados. Diz-se que um *software* falhou quando ele não cumpriu o que foi requisitado. Essas falhas podem estar relacionadas a requisitos mal elaborados ou até mesmo a falta do mesmo, impossibilitando a sua implementação (PFLEEGER, 2004).

Os testes de *software* devem ser persuasivos e eficazes aos desenvolvedores e clientes, pois, desta forma, segundo SOMMERVILLE (2007), o *software* estará em condições de uso, e atingirá uma qualidade de uso operacional.

Levando em conta que o analista projeta um sistema para ser o mais eficaz possível, em contrapartida ele também projeta os testes para que sejam encontrados erros críticos. Parece um absurdo, mas de fato, o teste de *software* representa uma forma de manipular o sistema a falhas. É da essência dos programadores construir softwares e testar é um trabalho que requer pôr de lado essa característica PRESSMAN (2009).

Para RIOS (2010), apesar do teste identificar a maioria dos *"bugs"* no programa, não se pode afirmar que o mesmo está isento de falhas. MYERS (2012), define que cobrir o *software* totalmente com testes não é uma tarefa fácil, pois há várias entradas a serem testadas e inúmeras possibilidades.

O grande problema para os testadores é justamente identificar as maiorias das probabilidades de combinações de entrada que são inseridas, já que poucas pessoas têm experiência neste assunto (MOLINARI, 2008).

Há várias maneiras de se testar um *software*, porém existem alguns métodos que são mais utilizados. Pode-se destacar o teste de unidade, teste de integração e teste de sistema. O teste de unidade é o primeiro a ser aplicado, e verifica o código fonte do sistema. Em seguida vem o teste de integração que vai verificar a integração entre as unidades, e, por último, o teste de sistema que verifica o desempenho do *software*, capacidade de carga, agilidade e autonomia (PRESSMAN, 2009).

Para combater os desafios associados a economia dos testes, de acordo com MYERS (2012), define-se uma estratégia antes de iniciar o teste propriamente dito. Uma das principais técnicas de teste são os Testes Funcionais, que incluem os Testes de Caixa Preta e Caixa Branca.

O teste caixa preta é um tipo de teste em que os colaboradores não têm conhecimento do conteúdo interno. A parte importante neste caso é o conteúdo externo. Segundo MALDONADO (2007), informações são produzidas de acordo com as entradas inseridas. MECENAS and OLIVEIRA (2005) descreve que nesta etapa deve-se testar todas as entradas e não se preocupar com o código fonte, ou seja, a linguagem de programação neste caso é irrelevante para aplicar o teste caixa preta. Normalmente este tipo de teste visa ser o último estágio da atividade de testes.

O teste de caixa branca é uma abordagem no qual se aplica o teste em nível de componente. Neste caso o ponto chave é a conhecimento interno das funcionalidades do sistema. Domínio da linguagem de programação no qual foi desenvolvida as funcionalidade é pré-requisito para aplicar o teste caixa branca PRESSMAN (2009). Este trabalho terá como base a estrutura do Teste Funcional, que compreende basicamente em avaliar as

funcionalidades do *software* por um âmbito externo.

2.2 Divisão dos Testes

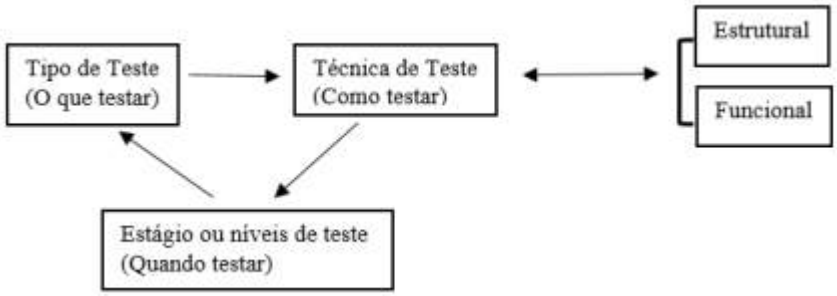


Figura 2.1: Figura Modelo das Fases do Teste.
Fonte: RIOS (2010).

Como mostra a Figura 2.1, o teste está dividido em:

- **Tipos de teste**: Existem diversos tipos de teste. Cada um tem funções específicas a cada etapa de desenvolvimento do projeto de acordo a necessidade. Alguns exemplos são MOLINARI (2008):
  - **Testes funcionais**: Testa de acordo com os requisitos.
  - **Teste de performace**: Testa o desempenho submetendo o sistema a cargas e obtendo dados(tempo) de sua performace.
  - **Teste de usabilidade**: Focado no grau de prática do usuário, *layout*, interface.
  - **Testes de unidade**: Testa partes isoladas, componentes ou classes.
  - **Testes integração**: Garantir que um ou mais componentes funcionem adequadamente em conjunto
  - **Testes de segurança**: Garantir que as informações só possam ser acessadas por quem tem de fato o direito.
  - **Testes de regressão**: Teste tudo ou uma parte novamente sempre que for inseridas novas funcionalidades.
  - **Testes de aplicações *web***: Caracteriza-se em testar *software* para *web* e integrado ao seu ambiente.
- **Níveis de teste**: Aponta qual rumo que o teste tomará.
  - **Sistema**: Testa-se o sistema como um todo, após a validações de todos os teste já executados.

  - **Aceitação**: Teste final antes de ir para o ambiente de produção. Neste teste verifica-se se o projeto atende as necessidades inicialmente propostas ao negócios. Este teste é feito pelos usuários.

- **Técnicas de teste**: Uma classificação quanto ao tipo de teste que se deseja fazer.
  - **Caixa Branca**: É aplicado na estrutura do projeto, neste caso, o código fonte.
  - **Caixa Preta**: Baseado nas funcionalidades, neste caso é o inverso do teste caixa branca, esta técnica não tem a visão do código fonte.

2.3 Teste Funcional

No teste funcional há dois passos principais: determinar as funções que o *software* deve realizar e gerar casos de teste com possibilidade de verificar essas funções. Na verdade, não somente as funções, mas também pode-se testar pequenas partes dos componentes como por exemplo uma DLL, já que o teste funcional testa funções de um sistema, mas nada impede de olhar a nível de componente, ou seja, a estrutura do sistema MALDONADO (2007). De acordo com PRESSMAN (2009), pode-se descobrir ausência de funções, erros de interface, estrutura, desempenho, inicialização e término.

2.4 Técnica de Teste

2.4.1 Caixa Branca

O teste de caixa branca está interessado na estrutura interna do sistema. Uma faixa do sistema é identificada e testada. Para este tipo de teste o conhecimento na linguagem que foi desenvolvido o *software* se faz necessário. Este teste poderá ser usado para complementar o teste de caixa preta para garantir uma maior cobertura (RIOS, 2010).

2.4.2 Caixa Preta

Esta técnica refere-se ao teste de um item do *software* sem o conhecimento de seu funcionamento interno e de como é executado. Não se tem a visão interna das funcionalidades, somente as especificações que descrevem o que o *software* deve fazer baseado nas atividades e resultados esperados. Conforme MOLINARI (2008), existem algumas técnicas específicas para teste funcionais. Entre elas pode-se destacar o teste de *Particionamento por Equivalência*, o teste *Análise de Valor Limite* e teste de *Tabela Verdade*.

2.4.2.1 Particionamento de Equivalência

Para esta técnica a ideia é dividir as entradas válidas e inválidas em grupos da mesma natureza, por exemplo: um campo pode receber valores inteiros entre 1235 a 1870, neste

caso todos os valores neste conjunto são válidos, e os valores inválidos são aqueles maiores que 1870 e menores que 1235. Pode-se aplicar em qualquer nível de teste a partir da seleção dos casos de teste de forma intuitiva. Esta técnica permite diminuir a quantidade de casos de testes. Figura 2.2.

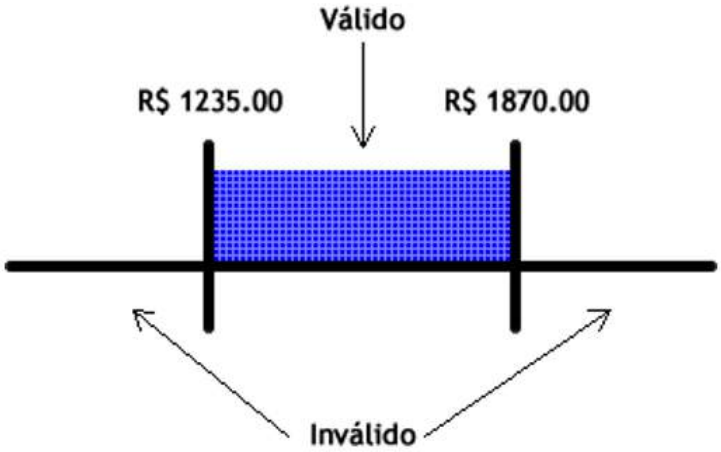


Figura 2.2: Figura Particionamento de Equivalência
Fonte: MOLINARI (2008)

2.4.2.2 Análise de Valor Limite

Este método costuma ser uma extensão da técnica de particionamento por equivalência, pois acredita-se que as falhas podem estar nos limites do particionamento, por exemplo: quando um caso de teste é projetado um valor de limite mínimo e um valor de limite máximo são definidos, desta forma é possível delimitar os valores válidos de fronteira, na Figura 2.3 representa uma classe onde um valor mínimo 100 e um valor máximo de 1000. Os valores válidos para essa classe são os números 99, 100 e 101 para valores mínimos e para os valores máximos são os números 999 , 1000 e 1001.

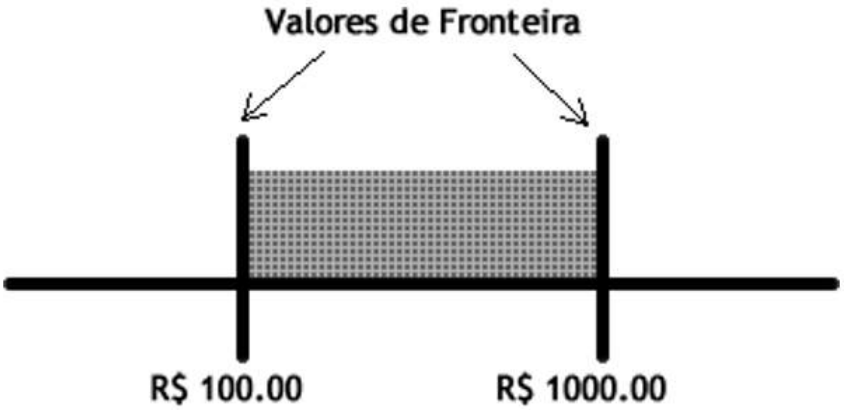


Figura 2.3: Figura Análise de Valor Limite
Fonte: MOLINARI (2008)

2.4.2.3 Tabela Verdade

É uma técnica no qual utiliza-se uma tabela contendo condições, regras e ações. Essas informações são originada de decisões lógicas contidas no sistema. Pode ser aplicada a qualquer momento e limita-se as regras de negócios. Portanto, se houver muitas regras, consequentemente o número de casos de teste também aumentará. Na Figura 2.4 demonstra um pequeno exemplo da tabela verdade onde há duas condições, quatro regras e conforme essa premissas são atendidas as ações são aplicadas.

	Regra 1	Regra 2	Regra 3	Regra 4
Condicões				
Casado(a)	Sim	Sim	Não	Não
Carteira +1 ano	Sim	Não	Sim	Não
Acões				
Desconto (R$)	60	25	50	0

Figura 2.4: Figura Tabela de Decisão
Fonte: próprio autor

2.5 Planejamento de Teste

Compreendendo os conceitos anteriormente citados, aplica-se então um Planejamento de Testes, que é uma estrutura com indicações técnicas destinadas a orientar a equipe de teste, bem como tempo e trabalho que serão necessários, e se for preciso, alterar segundo as especificações do projeto. O plano de teste compreende todas as informações e rotinas necessárias para a execução das tarefas (MOLINARI, 2008).

2.6 Plano de Teste

Os projetos podem ser grandes, pequenos ou médios. Dessa forma o plano de teste vai depender do tamanho do projeto. De acordo com RIOS (2010), podem ter vários planos de teste para um único projeto, mas, se o projeto for relativamente pequeno, somente um plano de teste será necessário. E, nestes planos classificam-se os níveis do teste, seja ele unitário, integração, sistema ou aceitação. De acordo KOSCIANSKI (2007), o plano de teste é um documento que serve como mediador entre toda a equipe e fornece uma estrutura padrão a ser seguida.

2.7 O Padrão IEEE 829 e Seus Documentos Relacionados

Este documento visa estabelecer uma organização textual, ou seja, um formato padrão que servirá como ponto de partida a ser utilizado por uma empresa a fim de organizar suas atividades, além de manter uma boa comunicação no âmbito da equipe. Este modelo apresenta 8 documentos divididos em 3 classes (IEEE-829, 2008);

- **Plano de teste**: Descreve as funcionalidades a serem testadas, o tempo reservado para execução dos testes, os objetivos e os responsáveis.
  - Documento criado durante o planejamento do teste e serve como um guia de referência.
- **Especificação de teste** : Descreve o projeto de teste e seus casos de teste relacionados, bem como as técnicas utilizadas.
  - Projeto de teste
  - Caso de teste
  - Procedimento
- **Relatórios** : Relata informações sobre os resultados obtidos nos procedimentos de testes executados.

  - Estado de testes
  - Log de testes
  - Incidentes
  - Final

2.8 Caso de Teste

O *caso de teste* descreve o que se precisa testar, investigando minuciosamente os requisitos de testes. Segundo MOLINARI (2008), um caso de teste poderá se referenciar a mais de um requisito de teste e espera-se que haja somente um conjunto de passos de teste para um caso de teste. No ponto de vista de RIOS (2010), a intenção do caso de teste é indicar uma unidade de teste que será controlada pelo testador, seja de forma manual ou automática. O caso de teste é composto por:

- Introdução
  - **Identificador do documento**: código que identifica o documento.
  - **Escopo**: o que este documento está se referindo.
  - **Referências**: qualquer informação que sirva de referência no uso de artefatos para caso de teste.
  - **Contexto**: informações que indica o que não foi coberto por outras seções deste documento.
  - **Notas para descrição**: detalhe e identificação de artefatos utilizados.
- Detalhe
  - **Identificador do caso de teste**: um código para identificar o documento.
  - **Objetivos**: qual o objetivo do caso de teste ou grupo de teste.
  - **Especificações de entrada**: quais as origens de entrada ou qualquer outra informação relevante.
  - **Especificações de saída**: o que será esperado.
  - **Ambiente**: especificar a necessidades dos ambientes para execução do teste.
  - **Requisitos ou procedimentos**: qualquer outro requisito necessário para que o caso de teste possa ser executado.
  - **Dependências entre caso de teste**: se depende de outro caso de teste.
- Global
  - **Glossário**: principais termos usados.
  - **Procedimento de alterações e histórico de alterações**: registrar informações feitas e quem as fez.

2.9 Ferramentas e Linguagens

2.9.1 C#

Para a execução deste estudo foi utilizada a Linguagem de programação *C# (lê-se C sharp)*, uma linguagem orientada a objeto e fortemente tipada. Sua sintaxe é muito parecida com *C/C++* e *Java*. *C#* está incorporado a IDE (Ambiente de Desenvolvimento Integrado,) da Microsoft conhecida como *Visual Studio*. A empresa é conhecida por comercializar seus *softwares*, mas mantêm uma versão gratuita conhecida como *Visual Studio Community*.

O arquivo de instalação se encontra disponível no seguinte site:

- *https://www.visualstudio.com/pt-br/vs/community/*

O *Visual Studio* atualmente está na versão 2017.

2.9.2 Banco de Dados

Para este trabalho foi utilizado o sistema de gerenciamento de banco de dados *MySQL*. Segundo o site MYSQL (2016), ele é um banco de dados rápido e multiusuários. É um *software* bastante famoso na comunidade *web* por suportar várias plataformas e várias linguagens de programação. *MySQL* é uma marca registrada da *Oracle Corporation*, possui licença dupla: o usuário pode optar em utilizar como um produto *open source* sob os termos GNU *General Public Licence*, ou ainda se preferir poderá optar em comprar uma licença comercial.

2.10 Revisão do Capítulo

Neste Capítulo, foram apresentadas a fundamentação teórica do projeto desenvolvido. Também foram exibidas as tecnologias utilizadas: linguagem de programação *C#*, sistema de gerenciamento de banco de dados *MySQL*.

Capítulo 3

Trabalhos Correlatos

Neste capítulo será exposto alguns softwares correlatos à nossa ferramenta existentes no mercado e demonstrar onde o estudo relaciona-se neste cenário.

3.1 Ferramentas Similares

Ao longo do estudo para o desenvolvimento da ferramenta proposta, verificou-se que há vários *sotwares* para contribuir na manipulação dos testes. Estas ferramentas estão relacionadas a teste funcionais para *web*, dentre os quais podemos citar:

- *Selenium* - http://seleniumhq.org
  - Um *framework open source* para *Java* que integra os principais *web browsers*, utilizado para teste automatizados manipulando tags de html.
- *Watir* - http://wtr.rubyforge.org
  - É uma estrutura de teste de *web*, código aberto, gratuita criada com *Ruby*. Seus teste também são realizados em *Ruby*.
- *BadBoy* - http://www.badboy.com.au
  - Ferramenta para automação de teste com vários recursos importantes, como relatórios, gráficos e geração de *scripts* MOLINARI (2008). Figura 3.1

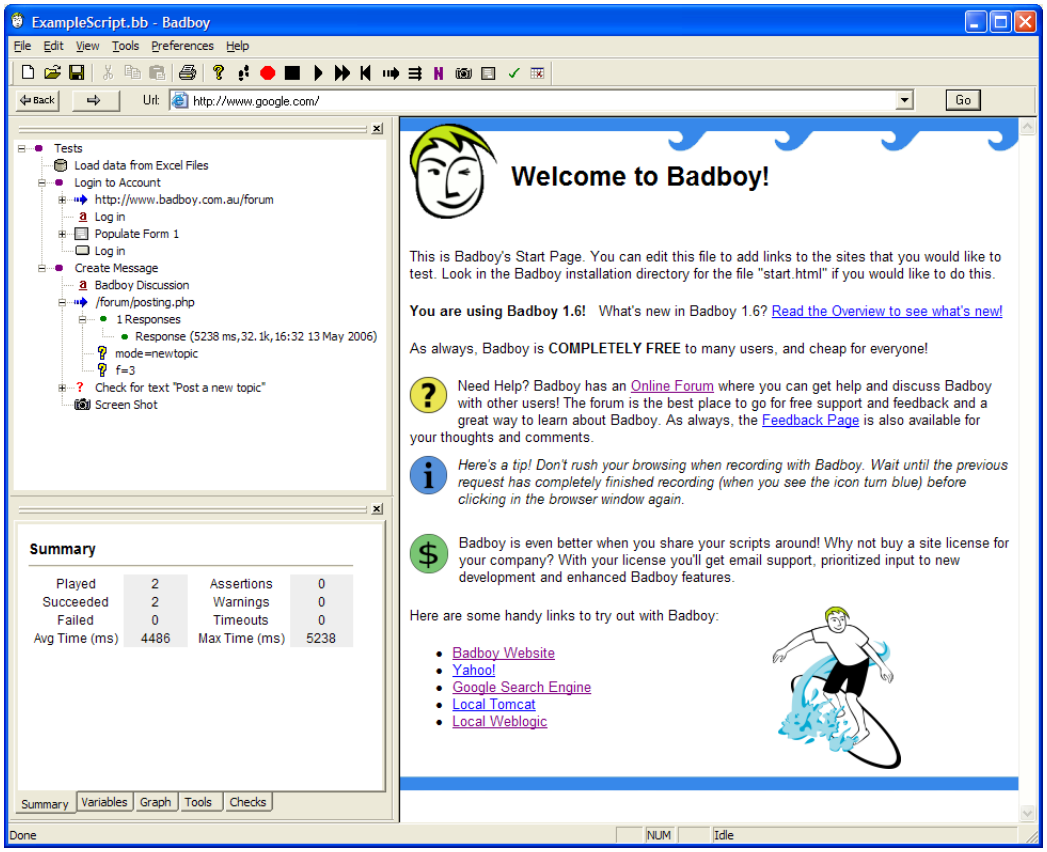


Figura 3.1: Figura Tela da ferramenta BadBoy
Fonte: BadBoy.

- *actiWATE* - https://www.actimind.com/actiwate.html
  - O actiWATE é fundamentado em *Java* e automatiza processos de testes através de *scripts* em aplicações web, semelhante a outros *frameworks* como *HttpUnit* e *HtmlUnit*. Figura 3.2 demonstra o funcionamento do framework. Módulo de escrita de teste, simulação de navegador e ação com base no script com suporte a protocolos HTTP e HTTPS.

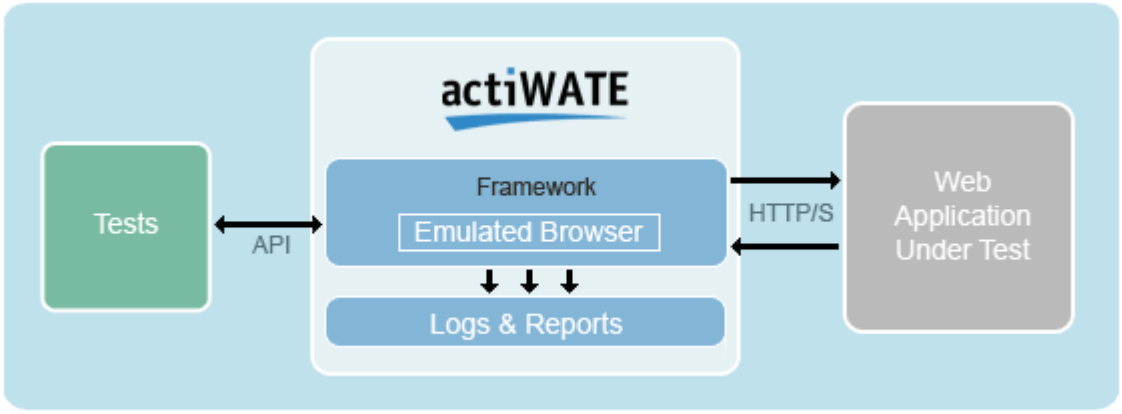


Figura 3.2: Figura Framework actiWATE
Fonte: actiWATE.

- *Canoo WEBTest* - http://WEBtest.canoo.com
  - De acordo com o site WEBtest a ferramenta é *open source*, livre, utilizada para automação de testes.
- *RTH-Turbo* - https://code.google.com/archive/p/rth-turbo/
  - Conforme MOLINARI (2008), o RTH-Turbo é uma ferramenta *open source* para planejamento de testes.

Os projetos citados têm características semelhantes em produzir testes automatizados, testes de unidade entre outros. Porém, percebeu-se que as ferramentas pesquisadas não faziam a cobertura da fase de elaboração dos testes. Elas possuem semelhanças, nas quais, os usuários incluem as informações a serem testadas. Dessa forma, poderá surgir erros na elaboração dos testes.

A ferramenta RTH-Turbo é a que mais se assemelha com a nossa ferramenta no quesito planejamento de teste, porém o projeto foi descontinuado. Essa informação foi obtida através do Leonardo Molinari, autor do livro Teste de Software Funcionais, diretamente através de e-mail em anexo A.

A concepção deste estudo é justamente dar suporte no que deverá ser testado, proporcionando uma melhor cobertura com o mínimo de testes a serem realizados. Para representar o panorama da organização das fases dos teste, a Figura 3.3 demonstra duas fases. A fase de elaboração onde se desenvolve documentos para o plano de teste, o caso de teste e as técnicas de teste. A fase de aplicação dos teste e resultados dos testes. As ferramentas como Selenium, Watir, BadBoy, actiWATE e Canoo WEBTest estão na fase de aplicação tratando o teste como um todo, onde o usuário deve informar o que deverá ser testado. O projeto desenvolvido neste estudo enquadra-se na fase de elaboração, especificamente na criação de documentos para casos de testes e técnicas teste de forma automática, dando ao usuário relatórios do que deverá testar.

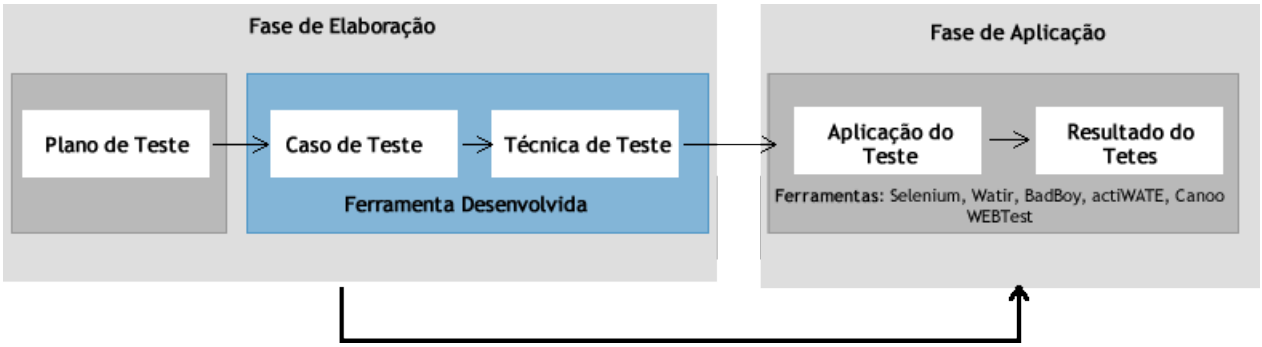


Figura 3.3: Figura Panorama dos testes
Fonte: próprio autor

3.2 Revisão do Capítulo

Neste capítulo apresentou-se algumas ferramentas de automatização de caso de teste funcionais, bem como o ambiente em que elas atuam e em qual fase de teste se relacionam. Demonstrou-se em qual fase de teste a ferramenta desenvolvida neste estudo está relacionada no ambiente de teste, tal como sua finalidade.

Capítulo 4

Metodologia e Ferramenta

Neste capítulo apresenta-se a metodologia para a construção da ferramenta para caso de teste funcional, a qual foi desenvolvida com objetivo de auxiliar o desenvolvedor na aplicação de teste.

4.1 Ferramenta Para Geração de Caso de Teste Funcional

O programa desenvolvido tem como escopo ser uma ferramenta para estabelecer características de testes funcionais para uma documentação que auxiliará os testadores na coleta de dados para serem aplicados em formulários de sistemas. O sistema disponibiliza primeiramente uma ação para o Cadastro do Projeto e em seguida uma função para o Cadastro dos Requisitos. Dentro da funcionalidade de requisitos, o usuário informa quais são os campos e seus tipos (número, string, data, etc) do formulário. Também é informado o tamanho máximo do valor em um campo (opcional). O sistema deve gerar um relatório com os possíveis casos de teste. A ideia é facilitar ao usuário, informando o número mínimo de testes funcionais que possa abranger o requisito a ser examinado.

4.1.1 Tipos de Dados

No levantamento para este estudo, classificamos os tipos de dados em 6 grupos distintos: Inteiro, Caractere, Data, Real, Time e Boolean. Logicamente estes tipos de dados são baseados em conceitos gerais uma vez que, cada banco de dados possui sua própria nomenclatura para os tipos de dados que operam. Na Tabela 4.1 demonstra qual tipo de dados é equivalente para cada grupo em seus respectivos bancos de dados. Dessa forma, é possível organizar uma estrutura hábil para aplicar as técnicas de testes propostas na seção 2.4.2.

Tabela 4.1: Tabela DataTypes.

Sql Server	**MySQL**	**PostgreSQL**
Caractere		
CHAR(n)	CHAR(n)	CHAR(n)
NCHAR(*n*)	NCHAR(*n*)	CHAR(*n*)
NTEXT	LONGTEXT	TEXT
NVARCHAR(*n*)	NVARCHAR(*n*)	VARCHAR(*n*)
NVARCHAR(*max*)	LONGTEXT	TEXT
TEXT	LONGTEXT	TEXT
Inteiro		
BIGINT	BIGINT	BIGINT
BINARY(n)	BINARY(n)	BYTEA
INT, INTEGER	INT, INTEGER	INT, INTEGER
SMALLINT	SMALLINT	SMALLINT
Boolean		
BIT	TINYINT	BOOLEAN
Date		
DATE	DATE	DATE
DATETIME	DATETIME(3)	TIMESTAMP(3)
SMALLDATETIME	DATETIME	TIMESTAMP(*0*)
Real		
DECIMAL(p,s)	DECIMAL(p,s)	DECIMAL(p,s)
DOUBLE PRECISION	DOUBLE PRECISION	DOUBLE PRECISION
FLOAT(p)	DOUBLE	DOUBLE PRECISION
MONEY	DECIMAL(*15,4*)	MONEY
NUMERIC(*p,s*)	NUMERIC(*p,s*)	NUMERIC(*p,s*)
REAL	REAL	REAL
SMALLMONEY(*n*)	DECIMAL(*6,4*)	MONEY
Time		
TIME(*p*)	TIME(*p*)	TIME(*p*)
TIMESTAMP	BINARY(*8*)	BYTEA(*n*)

A ferramenta deverá ser usada por um público que seja capaz de analisar os resultados por ela gerados. O foco no primeiro momento é liberar o programa para pessoas com formação para programação.

4.2 Apresentação das Técnicas de Testes Utilizadas

Foi utilizada três técnicas de teste: Tabela Verdade, Análise do Valor Limite e Particionamento por Equivalência. Essas técnicas foram empregadas de acordo com a tabela de tipos de dados apresentada na seção 4.1.1, levando em consideração as características semelhantes em cada banco de dados indicados na tabela 4.1.

4.2.1 Tipo: Caractere

Na Tabela 4.2, demonstra-se quais são os tipos de dados equivalentes para o campo CARACTERE dentro dos 3 (três) bancos de dados abordados na ferramenta.

Tabela 4.2: Tabela representando o tipo CARACTERE
.

Sql Server	**MySQL**	**PostgreSQL**
char(n)	char(n)	char(n)

4.2.1.1 Tabela Verdade

De acordo com a ideia proposta pelo autor MOLINARI (2008), na seção 2.4.2.3 definimos as seguintes condições para o teste de Tabela Verdade indicadas nas Tabelas 4.3 e 4.4, que estão relacionados ao tipo de campo CARACTERE.

Tabela 4.3: Tabela de condições: CARACTERE .

Condições	**Descrições**
Condição 1	Não aceitar caracteres especiais e números
Condição 2	Aceitar letras e letras com acentos
Condição 3	Não aceitar somente espaçamentos
Condição 4	Campo não pode ser nulo
Condição 5	Se Sql Server, o campo deve conter 255 caracteres. Se MySql, o campo deve conter x caracteres. Se PostgreSql, o campo deve conter x caracteres
Condição 6	O campo não pode conter somente 1 caractere

Na Tabela 4.3 há uma coluna com as condições enumeradas de 1 à 6, e uma outra coluna com as suas respectivas descrições contendo as informações para o tipo de campo

CARACTERE.

Tabela 4.4: Tabela de regras: CARACTERE

Regras	Verdadeiro	Falso
Condição 1 - regra 1	1- Teste com caracter especial 2- Teste com números	1- Teste com letras 2- Teste utilizando espaçamentos
Condição 2 - regra 2	1-Teste com letras 2-Teste com letras acentuadas	1-Teste com números 2-Teste com caractere especial 3- Teste utilizando espaçamentos
Condição 3 - regra 3	1-Teste preenchido com letras 2-Teste com letras acentuadas 3-Teste com números 4-Teste com caracteres especiais	1-Teste usando somente espaçamentos
Condição 4 - regra 4	1-Teste sem valor de entrada	1-Teste com letras 2-Teste com letras,acentuadas
Condição 5 - regra 5	1-O campo não pode conter somente 1 caractere	O campo contém mais que 1 caractere

Na Tabela 4.4 demonstra uma coluna de regras enumeradas de 1 à 5, cada condição definida na Tabela 4.3 está relacionada a uma regra. As ações estão relacionadas de acordo com as colunas nomeadas verdadeiro e falso como está representada na Tabela 4.4.

4.2.1.2 Particionamento por Equivalência

Conforme MOLINARI (2008), na seção 2.4.2.1 definimos as seguintes classes de equivalência para o tipo de campo CARACTERE, Tabela 4.5.

Tabela 4.5: Tabela Particionamento por equivalência: CARACTERE.

Intervalo de valores inválidos	Intervalo de valores válidos	Intervalo de valores inválidos
!...$...%...(....0...7...+...=...@	A.....D....J.....M.....R....T....Z	....a...h....j...n...p..r...t...

Na tabela 4.5 contém uma coluna que indica o intervalo de valores válidos permitidos e duas colunas denominadas intervalo de valores inválidos, no qual exibe os valores de entrada

que deverão ser rejeitados para o tipo de campo CARACTERE.

4.2.2 Tipo: Boolean

Na tabela 4.6, demonstra-se quais são os tipos de dados equivalentes para o campo BOOLEAN dentro dos 3 (três) bancos de dados abordados na ferramenta.

Tabela 4.6: Tabela representando o tipo: BOOLEAN.

Sql Server	**MySQL**	**PostgreSQL**
Bit	Boolean	Boolean

4.2.2.1 Tabela Verdade

Conforme disposto na seção 2.4.2.3 definimos as seguintes condições para o teste de Tabela Verdade relacionados ao tipo de campo BOOLEAN, Tabela 4.7

Tabela 4.7: Tabela de condições: BOOLEAN

Condições	**Descrições**
Condição 1	O campo não pode ser nulo

Na Tabela 4.7 há uma coluna com apenas uma condição e uma outra coluna com sua respectiva descrição contendo informações para o tipo de campo BOOLEAN

Tabela 4.8: Tabela regras: BOOLEAN

	Verdadeiro	Falso
Condição 1 -regra 1	1-Teste com valor,de entrada	1-Teste sem valor,de entrada

Na Tabela 4.8 demonstra uma coluna com apenas uma regra referenciando a condição definida na Tabela 4.7. As ações estão relacionadas de acordo como as colunas nomeadas verdadeiro e falso como está definido na Tabela 4.8.

4.2.3 Tipo: Inteiro

Na tabela 4.9, demonstra-se quais são os tipos de dados equivalentes para o campo Inteiro dentro dos 3 (três) bancos de dados abordados na ferramenta.

Tabela 4.9: Tabela representando o tipo:INTEIRO.

Sql Server	MySQL	PostgreSQL
INT	INT	INTEGER

4.2.3.1 Particionamento por Equivalência

Conforme disposto na seção 2.4.2.1 definimos as seguintes condições para o teste de Particionamento por Equivalência relacionados ao tipo de campo INTEIRO, Tabela 4.10.

Tabela 4.10: Tabela Particionamento por equivalência: Valores entre 100 e 200.

Intervalo de valores inválidos	Intervalo de valores válidos	Intervalo de valores inválidos
-3...4....6....18....45....60...99	100...120.....150.....190...200	201...450....900...1050...1090

Na tabela 4.10 contém uma coluna que indica o intervalo de valores válidos permitidos e duas colunas denominadas intervalo de valores inválidos, os quais informam os valores menores que o número 100 e valores maiores que o número 200 respectivamente.

4.2.3.2 Análise do Valor Limite

Conforme disposto na seção 2.4.2.2 definimos as seguintes condições para o teste de Análise do Valor Limite relacionados ao tipo de campo INTEIRO, Tabela 4.11.

Tabela 4.11: Tabela Análise do Valor Limite: Valores entre min=20 e Max=70.

Min=20	Meio>=Min E <=Max	Max=70
20	50	70
20	20	71
20	80	71
19	80	69
19	34	69
19	72	69
20	19	70
21	21	71

Na Tabela 4.11 há três colunas que indicam os valores de mínimo para ao número 20, a coluna meio que indica valores maiores que 20 e menores que 70 e a coluna máximo que indicam os valores máximos.

4.2.4 Tipo: Data

Na tabela 4.12, demonstra-se quais são os tipos de dados equivalentes para o campo Data dentro dos 3 (três) bancos de dados abordados na ferramenta.

Tabela 4.12: Tabela representando o tipo: DATA.

Sql Server	**MySQL**	**PostgreSQL**
DATE	DATE	DATE

4.2.4.1 Tabela Verdade

Conforme disposto na seção 2.4.2.3 definimos as seguintes condições para o teste de Tabela Verdade indicadas nas Tabelas 4.13 e 4.14 relacionados ao tipo de campo DATA.

Tabela 4.13: Tabela representando as condições para o tipo: DATA

Condições	**Descrições**
Condição 1	Aceitar somente números
Condição 2	A data não pode aceitar somente espaçamento
Condição 3	Quantidade de caractere = max permitida
Condição 4	Dois primeiros caracteres > 0 e <=31
Condição 5	Dois caracteres do meio >0 <=12
Condição 6	Datas válidas igual min. do banco e max. do banco. Caso não tenha sido definido valores mínimos e máximos.

Na Tabela 4.13 há uma coluna com as condições enumeradas de 1 à 6 e uma outra coluna com as suas respectivas descrições contendo informações para o tipo de campo DATE.

Tabela 4.14: Tabela representando as regas para o tipo: DATA

Regras	**Verdadeiro**	**Falso**
Condição 1 - Regra 1	1-Teste com números	1-Testar com qualquer caractere <>0...9
Condição 2 - Regra 2	1-Teste com números 2-Teste com espaçamento	1-Teste qualquer caractere <>0...9 2-Teste com espaçamento
Condição 3 - Regra 3	1-Teste com números 2-Quantidade de caractere = max.	1-Teste com qualquer caractere <>0..9 2-Teste com espaçamento 3-Teste quantidade de caractere <>max
Condição 4 - Regra 4	1-Teste com números 2-Teste com números >0 <=31, para representar o dia	1-Teste com qualquer caractere <>0..9 2-Teste com espaçamento 3-Teste quantidade de caractere <>max 4-Teste com números >31
Condição 5 - Regra 5	1-Teste com números 2-Teste com números >0 e <=12 para representar mês	1-Teste com qualquer caractere <>0..9 2-Teste com espaçamento 3-Teste quantidade de caractere <>max 4-Teste com números >31 , dd 5-Teste com números >12, MM
Condição 6 - Regra 6	1-Teste com números 2-Quantidade de caractere = max. 2-Teste com números >0 e <=31 para dia. 3-Teste com números >0 e <=12 para mês	1-Teste com qualquer caractere <>0..9 2-Teste com espaçamento 3-Teste quantidade de caractere <>max 4-Teste com números >31 , dd 5-Teste com números >12, MM

Na Tabela 4.14 demonstra uma coluna de regras enumeradas de 1 à 6, cada condição definida na Tabela 4.13 está relacionada a uma regra. As ações estão relacionadas de acordo como as colunas nomeadas verdadeiro e falso, exibidas na Tabela 4.14.

4.2.4.2 Particionamento por Equivalência

Conforme disposto na seção 2.4.2.1 definimos as seguintes condições para o teste de Particionamento por Equivalência relacionados ao tipo de campo Data, Tabela 4.15.

Tabela 4.15: Tabela Particionamento por equivalência: DATA

Intervalo de valores inválidos	**Intervalo de valores válidos**	**Intervalo de valores inválidos**
!...$...%...(....%...*...+...-..)	0.....3....6.....9	: < @ A d j

Conforme disposto na Tabela 4.15 definiu-se duas colunas que representam os intervalos de valores inválidos que são entradas rejeitadas e uma outra coluna que representa os intervalos de valores válidos que são as entradas permitidas.

4.2.4.3 Análise do Valor Limite

Conforme disposto na seção 2.4.2.2 definimos as seguintes condições para o teste de Análise do Valor Limite relacionados ao tipo de campo DATA, Tabela 4.16

Tabela 4.16: Tabela Analise do Valor Limite: Valores entre 01/01/1900 a 31/12/2020

Min=01/01/1900	**Meio>=Min E <=Max**	**Max=31/12/2020**
01/01/1900	01/01/2017	31/12/2020
01/01/1900	01/01/1900	01/01/2021
01/01/1900	01/01/2021	01/01/2021
01/01/1899	01/01/1900	01/01/1900
01/01/1899	27/05/1974	27/05/1974
01/01/1899	01/01/2021	27/05/1974
01/01/1900	01/01/2021	27/05/1974
01/01/1899	27/05/1974	01/01/2021

Na Tabela 4.16 a coluna nomeada Min indica os valores mínimos para serem testados, na coluna nomeada Meio indica os valores maiores que 01/01/1900 e os valores menores que 31/12/2020 para testar, já a coluna nomeada Max indica os valores máximos em relação ao valor 32/12/2020.

4.2.5 Tipo: Real

Na tabela 4.17, demonstra-se quais são os tipos de dados equivalentes para o campo Real dentro dos 3 bancos de dados abordados na ferramenta.

Tabela 4.17: Tabela representando o tipo: REAL

Sql Server	MySQL	PostgreSQL
REAL	REAL	REAL

4.2.5.1 Tabela Verdade

Conforme disposto na seção 2.4.2.3 definimos as seguintes condições para o teste de Tabela Verdade indicados nas Tabelas 4.18 e 4.19 relacionados ao tipo de campo REAL.

Tabela 4.18: Tabela representando as condições para o tipo: Real

Condições	Descrições
Condição 1	Aceitar somente números
Condição 2	A campo não pode aceitar somente espaçamento
Condição 3	Pode aceitar número negativos S/N

Na Tabela 4.18 há uma coluna com as condições enumeradas de 1 a 3 e uma outra coluna com as suas respectivas descrições contendo informações para o tipo de campo REAL

Tabela 4.19: Tabela representando as regras para o tipo: Real

Regras	Verdadeiro	Falso
Condição 1 - Regra 1	1-Teste com números	1-Testar com qualquer caractere <>0...9
Condição 2 - Regra 2	1-Teste com números	1-Teste qualquer caractere <>0...9 2-Teste com espaçamento

continua na próxima página

Tabela 4.19 – continuação da página anterior

Regras	Verdadeiro	Falso
Condição 3 - Regra 3	Se: Aceita valores negativos 1-Teste com números	1-Teste com qualquer caractere <>0..9 2-Teste com espaçamento 3-Teste com valores < que o minimo
Condição 3 - Regra 4	Se: Não Aceita valores negativos 1-Teste com números	1-Teste com qualquer caractere <>0..9 2-Teste com espaçamento 3-Teste com valores < 0

Na Tabela 4.19 demonstra uma coluna de regras enumeradas de 1 a 4, cada condição definida na Tabela 4.18 está relacionada a uma regra. As ações estão relacionadas de acordo como a coluna denominada verdadeiro e a coluna denominada falso, como indicando na Tabela 4.19.

4.2.5.2 Particionamento por Equivalência

Conforme disposto na seção 2.4.2.1 definimos as seguintes condições para o teste de Particionamento por Equivalência relacionados ao tipo de campo REAL, Tabela 4.20.

Tabela 4.20: Tabela Particionamento por equivalência: Caracteres 0 a 9

Intervalo de valores inválidos	Intervalo de valores válidos	Intervalo de valores inválidos
!...$...%...(....%...*...+...-..)	0.....3....6.....9	: < @ A d j

Conforme disposto na Tabela 4.20 definiu-se duas colunas que representam os intervalos de valores inválidos que são entradas rejeitadas e uma outra coluna que representa os intervalos de valores válidos que são as entradas permitidas.

4.2.5.3 Análise do Valor Limite

Conforme disposto na seção 2.4.2.2 definimos as seguintes condições para o teste de Análise do Valor Limite relacionados ao tipo de campo REAL, Tabela 4.21.

Tabela 4.21: Tabela Análise do Valor Limite: Valores entre 1.000,00 a 5.000,00

Min=1.000,00	**Meio>=Min E <=Max**	**Max=5.000,00**
1.000,00	3.000,00	5.000,00
1.000,35	3.000,00	5.000,01
1.000,00	5.000,01	5.000,02
999,99	5.000,01	5.000,03
999,99	3.500,01	4.999,99
999,99	5.000,01	4.999,99
1.000,01	5.000,01	4.999,99
999,99	4.999,99	5.000,01

Na Tabela 4.21 há três colunas que indicam os valores de mínimos para o valor 1.000,00, a coluna meio que indica valores maiores que 1.000,00 e menores que 5.000,00 e a coluna máximo que indicam os valores máximos avaliados.

4.3 Modelo Entidade Relacionamento

A Figura 4.1 apresenta um esboço do diagrama de Entidades e Relacionamento (DER) desenvolvido neste projeto.

- Entidade [**usuario**] - Armazena os usuários cadastrados.
- Entidade [**Projetos**] - Armazena os Projetos que serão usados para o teste.
- Entidade [**Requisito**] - Armazena o requisito de um determinado projeto.
- Entidade [**ComporRequisito**] - Armazena as informações detalhadas dos requisitos.
- Entidade [**PlanoTeste**] - Armazena as informações sobre Plano de Testes Gerados.
- Entidade [**CasoTeste**] - Armazena as informações sobre Caso de Testes Gerados.
- Entidade [**TipoCampos**] - Armazena as informações sobre os tipos de campos relacionados com bancos.
- Entidade [**TabelaVerdade**] - Armazena as informações sobre as condições e regras da Técnica da Tabela Verdade.
- Entidade [**Banco**] - Armazena as informações sobre os bancos que a ferramenta suporta.

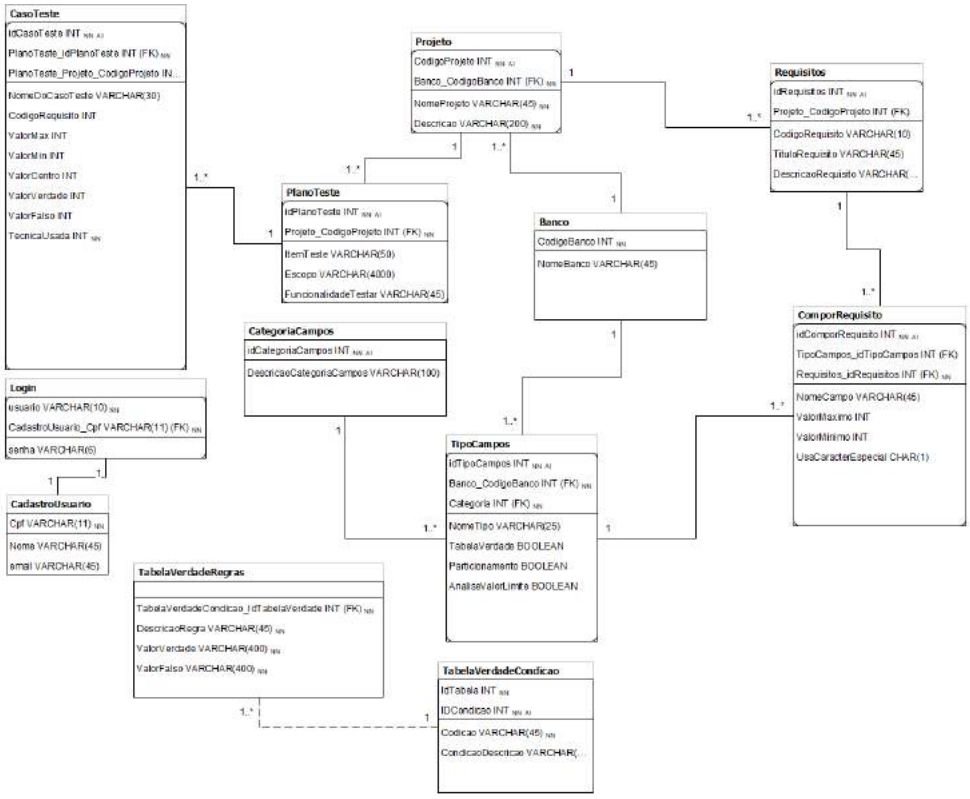


Figura 4.1: Figura Visão do modelo de entidades
Fonte: próprio autor

4.4 Revisão do Capítulo

Neste capítulo, foram apresentadas o desenvolvimento da metodologia das técnicas de teste baseando na fundamentação teórica. Também foram exibidas as tecnologias que serão utilizadas: linguagem de programação *C#*, sistema de gerenciamento de banco de dados *MySQL*.

Capítulo 5

A Ferramenta

Para o desenvolvimento foi utilizado a linguagem de programação C#, uma linguagem moderna e de fácil absorção, orientada a objeto e produtiva podendo inclusive ser usada para desenvolver Aplicativos Mobile para *Windows Phone*, *Android* e *IPhone IOS*, bem como para *web* e *Desktop*. Neste estudo optamos em desenvolver para *Desktop*. Toda a parte gráfica da ferramenta foi desenvolvida usando o *Windows Forms* que é uma *API* gráfica do conjunto de bibliotecas gerenciadas pelo *.NET Framework*. Os conceitos e orientação a objeto também foram aplicados no código de programação da ferramenta desenvolvida já que a linguagem utilizada nos permite tal abordagem. Também foi utilizado o banco de dados *MySql* para armazenamento dos dados. C# tem suporte a vários bancos de dados como: *PostgreSQL, Oracle, SQL Server, Access* e *MySql*.

5.1 Disponibilidade e Acesso à Ferramenta

Após o desenvolvimento, a ferramenta será disponibilizada no *Google Drive* através de um instalador. Podendo ser compartilhado através de uma solicitação por e-email.

5.1.1 Classes Utilizadas

Durante a fase de desenvolvimento da ferramenta, foram criadas classes para apoiar na programação. Na figura 5.1 expomos as mesmas, em seguida há uma breve explicação.

Estas classes são propriedades

- Classe [**Banco**] - Representa os bancos utilizados.
- Classe [**CasoTeste**] - Representa o caso de teste gerado.
- Classe [**ComporRequisito**] - Representa os itens do requisito.
- Classe [**PlanoTeste**] - Representa o plano de teste gerado.
- Classe [**Projeto**] - Representa o projeto gerado.

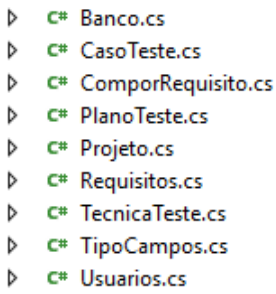


Figura 5.1: Figura Classes de domínio
Fonte: próprio autor

- Classe [**Requisitos**] - Representa o requisito do projeto.
- Classe [**TecnicaTeste**] - Representa as técnicas de testes.
- Classe [**TiposCamos**] - Representa os tipos suportados.
- Classe [**Usuários**] - Representa os usuários inseridos.

Apresentamos na Figura 5.2 as classes utilizadas para regras de negócios

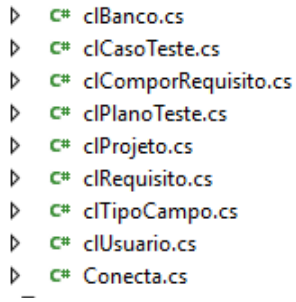


Figura 5.2: Figura Classes de regras de negócio
Fonte: próprio autor

- Classe [**clBanco**] - Utilizada para Inserir, Buscar, Excluir informações do Banco suportados.
- Classe [**clComporRequisito**] - Esta classe recupera, remove e atualiza os itens do requisito.
- Classe [**clPlanoTeste**] - Esta classe mantem e recupera informações do plano de teste do projeto.
- Classe [**clProjeto**] - Esta classe define a manipulação dos dados do projeto.
- Classe [**clRequisito**] - Esta classe conta com funcionalidades para manipulação dos requisitos do projeto.

- Classe [**clTiposCampo**] - Esta classe nos permite listar os tipos de dados.
- Classe [**clUsuário**] - Nesta classe estão as regras para Inserir, Atualizar, Buscar e Exluir usuários.
- Classe [**Conecta**] - Utilizada para criar uma conexão com o banco de dados.

A Figura 5.3 mostra a tela para acesso a ferramenta. Após efetuar o *login*, o usuário terá pleno acesso às suas funcionalidades.

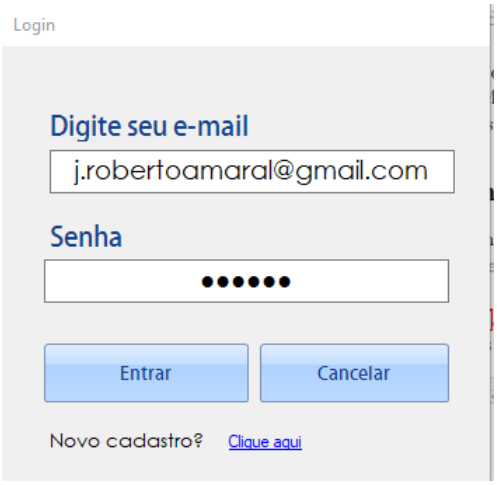


Figura 5.3: Figura Tela de acesso.
Fonte: próprio autor

Na Figura 5.4, exibimos um trecho do código da tela de *login* da Figura 5.3.

```
36      private void btnEntrar_Click(object sender, EventArgs e)
37      {
38          MySqlConnection cnn = new MySqlConnection("Persist Security Info = False; server = localhost; database = automatizedb; uid = root;");
39
40
41
42          try
43          {
44              cnn.Open();
45
46              StringBuilder sb = new StringBuilder();
47              sb.Append("SELECT * FROM login ");
48              sb.Append("WHERE usuario like '"+ txtLoginEmail.Text.ToString() +"' AND senha like '"+ txtLoginSenha.Text.ToString()+"'");
49              MySqlCommand cmd = new MySqlCommand(sb.ToString(),cnn);
50
51
52              MySqlDataReader dr;
53              dr = cmd.ExecuteReader();
54
55              if(!dr.HasRows)
56              {
57                  MessageBox.Show("Dados não conferem. Digite seu email e senha novamente", "Validação de Dados", MessageBoxButtons.OK, MessageE
58                  return;
59              }
```

Figura 5.4: Figura Trecho de código da Tela de acesso.
Fonte: próprio autor

Na linha 38, é feita conexão com o banco de dados da ferramenta e na linha 55 retorna o resultado e se obteve sucesso, libera o acesso ao usuário.

Na tela principal da ferramenta (Figura 5.5), são exibidas as funcionalidades disponibilizada ao usuário:

- Cadastro de Projetos
  - Usado para cadastrar, editar e excluir um projeto.

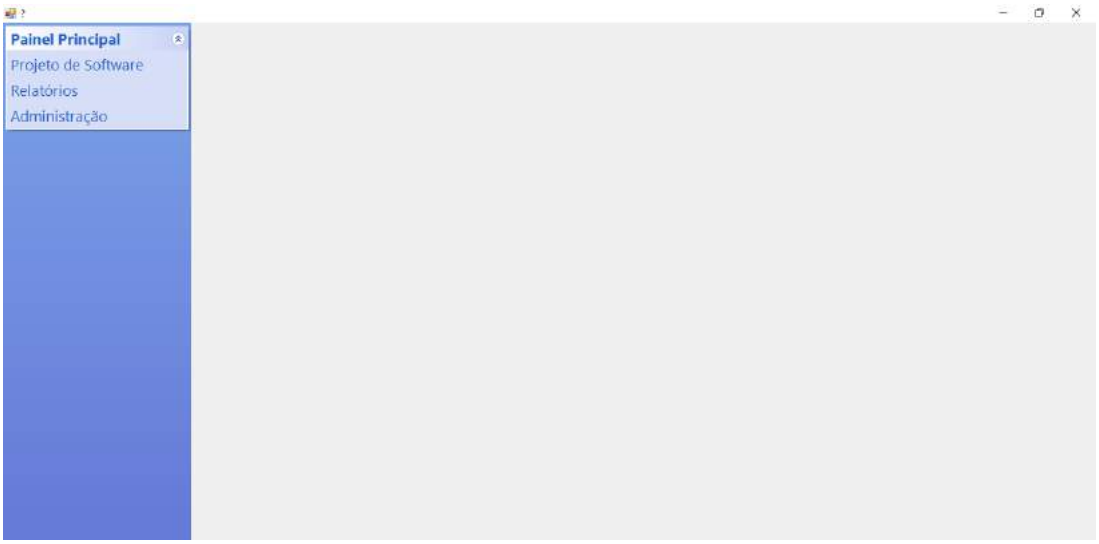


Figura 5.5: Figura Tela principal da ferramenta.
Fonte: próprio autor

Na tela de cadastro dos projetos, Figura 5.6, estão dispostos os seguintes campos:

- Pesquisar projeto
  - Usado para buscar um projeto já cadastrado.
- Nome do Projeto
  - Usado para registrar nome do projeto.
- Banco de Dados
  - Usado para descrever qual banco de dados será usando pela ferramenta
- Descrição do projeto
  - Usado para descrever com mais detalhes informações do projeto.

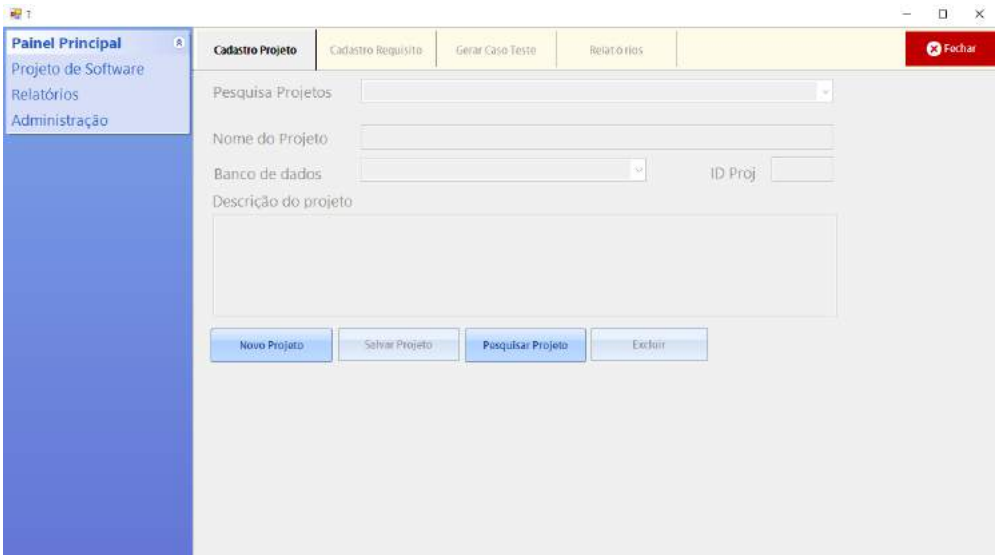


Figura 5.6: Figura Tela cadastro de projetos.
Fonte: próprio autor

Na Figura 5.7, é exposto o trecho do código para cadastro do projeto. Na linha 157 é aplicado o uso da classe *Projeto*, na linha 166 a classe retorna se a função para cadastrar obteve sucesso.

```
145     }
146
147     private void btnCadastrarProjeto_Click(object sender, EventArgs e)
148     {
149         //Faz uma validação para os campos nulos
150         if (string.IsNullOrEmpty(txtNomeProjeto.Text) || string.IsNullOrEmpty(cboBancos.Text) || string.IsNullOrEmpty(txtDescricaoprojeto.Text))
151         {
152             MessageBox.Show("Todos os campos devem estar preenchidos", "Atutomatize", MessageBoxButtons.OK, MessageBoxIcon.Warning);
153             return;
154         }
155
156         //Usando a classe Projeto para passar as propriedades dos campos
157         var proj = new Projeto();
158         proj.CodigoBanco = Convert.ToInt16(cboBancos.SelectedValue);
159         proj.NomeProjeto = txtNomeProjeto.Text;
160         proj.DescricaoProjeto = txtDescricaoprojeto.Text;
161
162
163         //Executa a Inserção do Projeto passando a classe Proeto como parametro
164         var Clproj = new Classes.clProjeto();
165
166         if (!Clproj.InsereProjeto(proj))
167         {
168             //Se Retornar Erro
169             MessageBox.Show("Não foi possivel efetuar o cadastro!", "Atutomatize", MessageBoxButtons.OK, MessageBoxIcon.Warning);
170         }
171         else
172         {
173             //Obteve sucesso na inserção
174             MessageBox.Show("Cadastro feito com sucesso!", "Atutomatize", MessageBoxButtons.OK, MessageBoxIcon.Information);
175             txtCodigoProjeto.Text = proj.CodigoProjeto.ToString();
176             EncheComboProjeto();
177         }
178     }
```

Figura 5.7: Figura Trecho do código para cadastrar o projeto.
Fonte: próprio autor

A Figura 5.8 representa o cadastro de requisitos dos projetos.

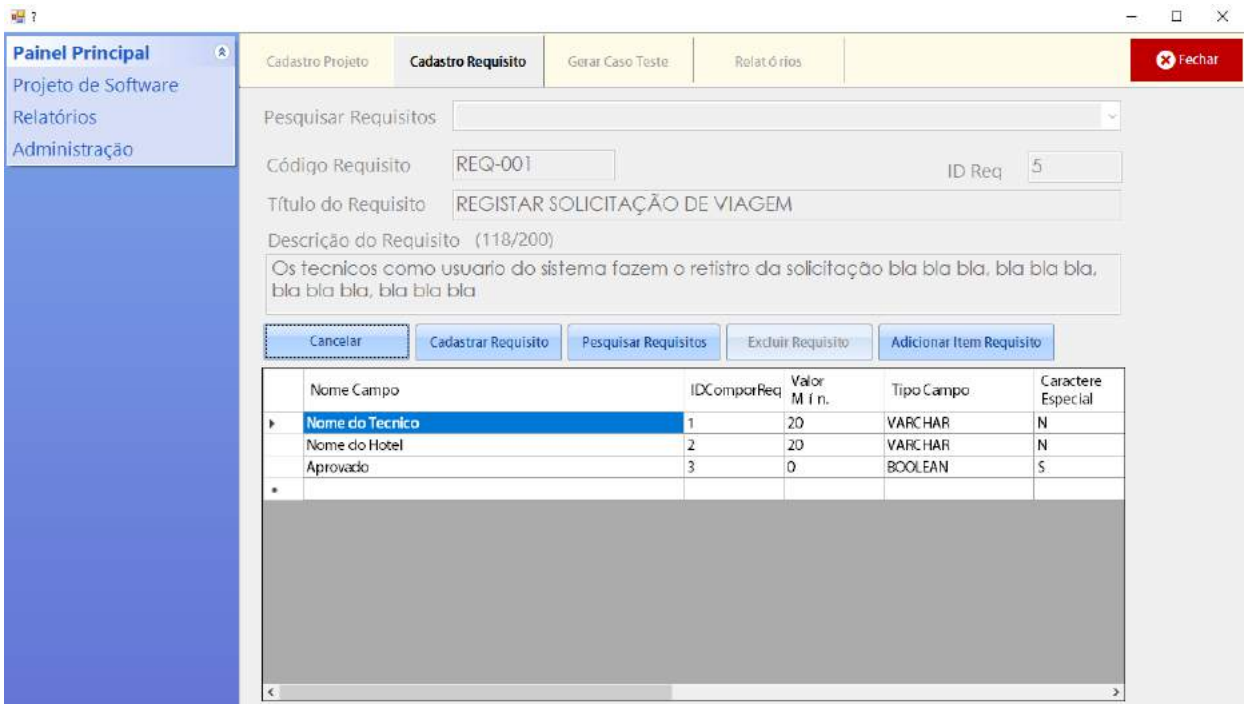


Figura 5.8: Figura Visão da tela de cadastro de requisitos
Fonte: próprio autor

Estarão dispostos nesta tela os seguintes campos:

- Código do Requisito - Informação que está condicionada ao requisito do sistema.
- Título do requisito - Informação descrita no requisito do sistema.
- Descrição do requisito - Especificação mais detalhada.
- Campos do formulário - Usado para descrever quais campos, tipo e tamanho serão testados.
- Botão Gravar - Faz a gravação dos dados no sistema.

Na Figura 5.9, representamos trecho do código para o cadastro dos requisitos na ferramenta. Na linha 190 é criada uma instância da classe *Requisitos*, logo em seguida na linha 198 é utilizado a classe *clRequisito* para utilizar a função de *InsereRequisito*.

```
180     private void btnCadastraReq_Click(object sender, EventArgs e)
181     {
182         //Faz uma validação para os campos nulos
183         if (string.IsNullOrEmpty(txtCodigoRequisito.Text) || string.IsNullOrEmpty(txtTituloRequisito.Text) || string.IsNullOrEmpty(txtDescricaoRequisito.Text))
184         {
185             MessageBox.Show("Todos os campos devem estar preenchidos", "Atutomatize", MessageBoxButtons.OK, MessageBoxIcon.Warning);
186             return;
187         }
188
189         //Usando a classe Requisitos para passar as propriedades dos campos
190         var req = new Requisitos();
191         req.Projeto_CodigoProjeto = int.Parse(txtCodigoProjeto.Text);
192         req.TituloRequisito = txtTituloRequisito.Text;
193         req.CodigoRequisito = txtCodigoRequisito.Text;
194         req.DescricaoRequisito = txtDescricaoRequisito.Text;
195
196
197         //Executa a Inserção do requisito passando a classe Requisitos como parametro
198         var Clreq = new Classes.clRequisito();
199
200         if (!Clreq.InsereRequisito(req))
201         {
202             //Se Retornar Erro
203             MessageBox.Show("Não foi possível efetuar o cadastro!", "Atutomatize", MessageBoxButtons.OK, MessageBoxIcon.Warning);
204
205         }
206         else
207         {
208             //Obteve sucesso na inserção
209             MessageBox.Show("Cadastro feito com sucesso!", "Atutomatize", MessageBoxButtons.OK, MessageBoxIcon.Information);
210             txtIdRequisito.Text = req.IdRequisito.ToString();
211             EncheComboRequisito();
212             btnAddItemReq.Enabled = true;
213             LockedUnlockedReq(false);
214             btnNovoCancelar.Text = "Novo Requisito";
215
216         }
```

Figura 5.9: Figura Trecho do código para cadastrar requisito
Fonte: próprio autor

Na Figura 5.10 apresentamos a tela de Cadastro de Item do Requisito. É importante salientar que esta opção só estará disponível quando houver um requisito devidamente cadastrado.

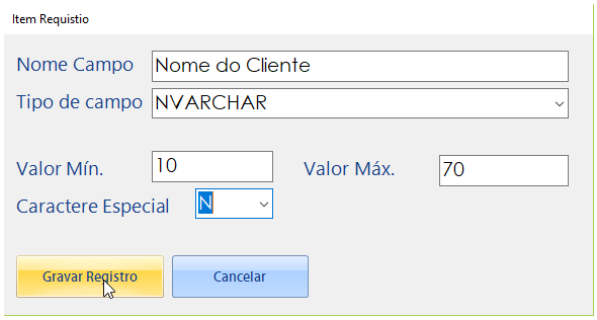


Figura 5.10: Figura Tela do Cadastro de Item do Requisito
Fonte: próprio autor

Na Figura 5.11 apresentamos trechos do código para Cadastrar os Itens do Requisito. Na linha 43 é criada uma nova instância da classe *ComporRequisito*, nesta classe estão as propriedade utilizada pela classe *clComporRequisito* situada na linha 52, que emprega a função *InsereItemRequisito*.

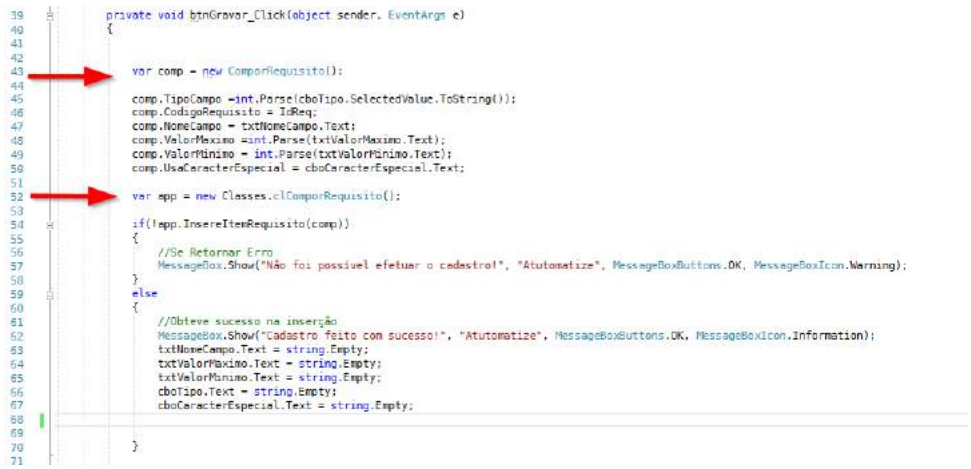


Figura 5.11: Figura Tela do Cadastro de Item do Requisito
Fonte: próprio autor

Em campos da tela, o usuário poderá optar pelo tipo de campo a ser testado. Os tipos disponíveis da ferramenta são: Caractere, Número Inteiro, Número Decimal, Data, Hora e Booleano. De acordo com o tipo do campo, a ferramenta apresentará quais os parâmetros necessários para a geração dos casos de teste conforme listado a seguir:

- Caractere: Nome do Campo, Tamanho, Caractere especial (sim/não).
- Número Inteiro: Nome do Campo, Tamanho Máximo (opcional), Tamanho Mínimo (opcional).
- Número Decimal: Nome do Campo, Tamanho Máximo (opcional), Tamanho Mínimo (opcional). Número de casas reais.
- Data: Nome do Campo, Data mínima, Data máxima.
- Hora: Nome do Campo, Hora mínima, Hora máxima.
- Booleano: Nome do Campo.

5.2 Revisão do Capítulo

Neste capítulo exibiu-se o projeto da ferramenta para a geração automática de casos de teste. Para a construção das telas foram utilizados componentes da *API* gráfica *Windows Forms*. Apresentou-se um diagrama (DER), com o objetivo de esclarecer as características da ferramenta. Além disso, descrevemos algumas telas desenvolvida para a ferramenta.

Capítulo 6

Resultados

6.1 Estudo de Caso: Locadora

A Empresa ABC é uma grande locadora de filmes e mantém um grande acervo de variados filmes. Com o crescimento de sua carteira de clientes, percebeu a necessidade de automatizar a empresa para ter o maior controle sobre seu estoque e melhor atender os consumidores. Para um melhor entendimento deste estudo de caso são apresentados a seguir a descrição dos requisitos funcionas bem como as tabelas que os relacionam.

- Requisitos Funcionais
  - Req-001 - Manter cadastro de cliente, Tabela 6.1
  - Req-002 - Manter usuários no sistema, Tabela 6.2
  - Req-003 - Manter filmes,, Tabela 6.3
  - Req-004 - Manter cadastro de gênero de filmes, , Tabela 6.4
  - Req-005 - Manter locação, Tabela 6.5
  - Req-006 - Manter Item de locação, Tabela 6.6

Tabela 6.1: Tabela REQ-001

Código	**Descrição**	
Req-001	Manter cadastro de clientes	
Usuário com perfil nível 1 poderá fazer o cadastro de novos clientes bem como editar suas informações para manter atualizado		
Nome campo	**Tipo**	**Descrição**
Número do Cpf	INT(11)	Informações do CPF do cliente
Nome do Cliente	VARCHAR(50)	Informação do nome do cliente
Telefone	INT(10)	Informação do telefone do cliente
Email	VARCHAR(50)	Informação do email do cliente
Data do Cadastro	DATE	Informação da data de cadastro
Data do Aniversário	DATE	Informação da data do aniversário do cliente

Tabela 6.2: Tabela REQ-002

Código	**Nome**	
REQ-002	Manter usuários no sistema	
Usuário com perfil nível 2 poderá fazer o cadastro de novos usuários bem como editar, inativar e bloquear seus acessos		
Nome Campo	**Tipo**	**Descrição**
Nome do Usuario	VARCHAR(50)	Armazena o nome do usuário
Email	VARCHAR(50)	Armazena o email do usuário
Senha	CHAR(10)	Senha do Usuário
Data do Ultimo Acesso	DATE	Data do último acesso

Tabela 6.3: Tabela REQ-003

Codigo	**Nome**	
REQ-003	Manter filmes	
Usuário com perfil nível 1 poderá realizar o cadastro de novos filmes, editar e consultar. Pré-condições: REQ-004		
Nome Campo	**Tipo**	**Descrição**
Código do Filme	INT	Informação do identificador do filme
Gênero do Filme	INT	Informação do identificador do gênero
Nome do Filme	VARCHAR(50)	Informação para o nome do filme
Valor da Diária	FLOAT	Informação sobre o valor da diária da locação
Data da Ultima Locação	DATE	Data do última locação

Tabela 6.4: Tabela REQ-004

Codigo	**Nome**	
REQ-004	Manter cadastro de gênero de filmes	
Usuários com Perfil nível 1, poderá inserir, consultar e editar informações do cadastro de gêneros.		
Nome Campo	**Tipo**	**Descrição**
Código do Gênero	INT	Informação do identificador do gênero
Nome do Gênero	VARCHAR(50)	Informação sobre o nome do gênero

Tabela 6.5: Tabela REQ-005

Codigo	**Nome**	
REQ-005	Manter locação	
A administração e usuário do sistema poderão criar, editar, consultar e cancelar locações pertinentes. Pré-condições: REQ-001, REQ-003		
Nome Campo	**Tipo**	**Descrição**
Código da Locação	INT(11)	Informação do identificador da locação
Cpf do Cliente	INT(10)	Informação do cpf do cadastro de cliente
Data da Saída	DATE	Informação da data que foi retirado
Data do Retorno	DATE	Informação para devolver o filme

Tabela 6.6: Tabela REQ-006

Codigo	**Nome**	
REQ-006	Item de Locação	
Mantém os dados dos itens da locação Pré-condições: REQ-005		
Nome Campo	**Tipo**	**Descrição**
Código da Locação	INT	Informação da Locação
Código do Filme	INT	Informação do Filme
Número de Dias	INT	Informação sobre diárias

6.2 Aplicando a Ferramenta

Nesta seção é feito o ensaio da aplicabilidade da ferramenta desenvolvido no estudo de caso locadora conforme relatado no seção 6.1.

Na Figura 6.1, exibimos a tela de Cadastro do Projeto, no qual demonstra o ensaio do cadastrando de um novo projeto.

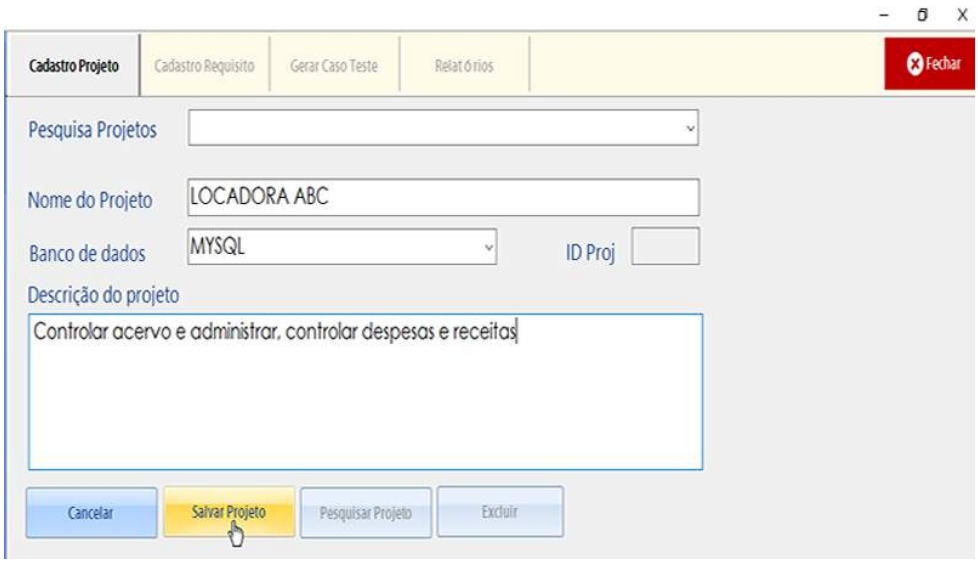


Figura 6.1: Figura Tela de Cadastro de Projeto
Fonte: próprio autor

Na Figura 6.2, exibimos a tela de Cadastro de Requisitos, no qual demonstra o ensaio do cadastramento do Req-001 - Manter cadastro de clientes.

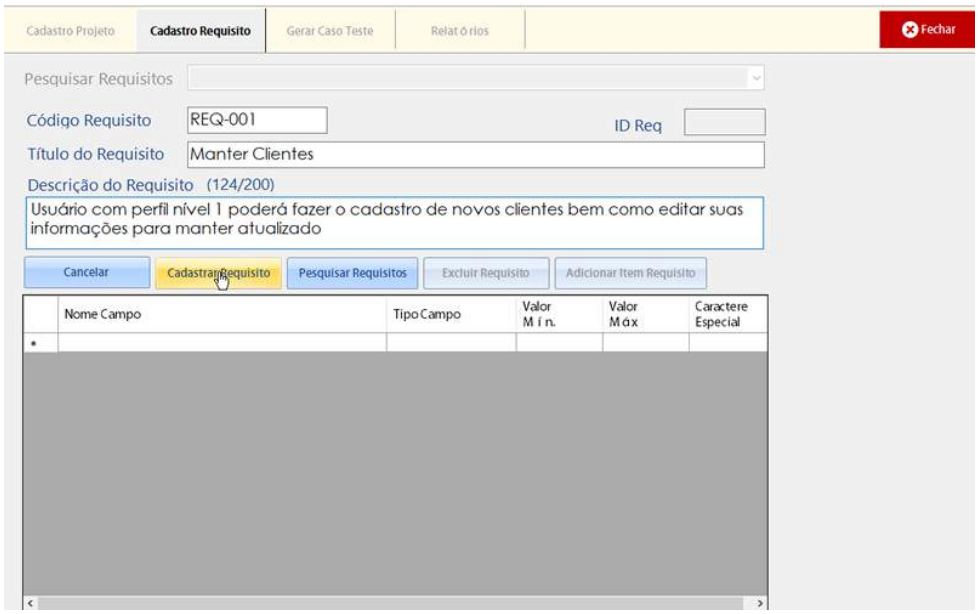


Figura 6.2: Figura Tela de Cadastro de Requisitos
Fonte: próprio autor

Na Figura 6.3, exibimos a tela de Cadastro dos Itens do Requisitos, no qual demonstra o ensaio do cadastramento dos itens do requisito Req-001 - Manter cadastro de clientes.

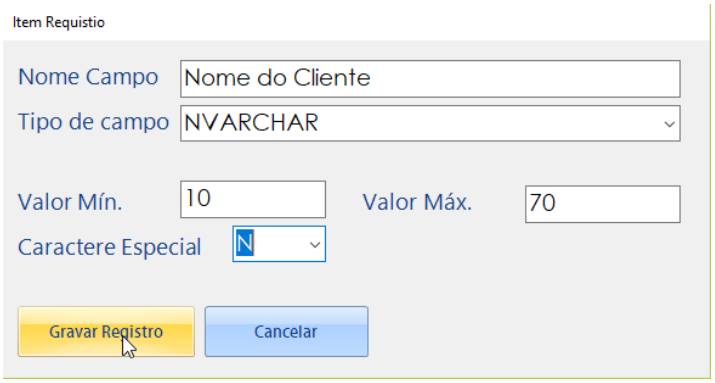


Figura 6.3: Figura Tela de Cadastro dos Itens do Requisitos
Fonte: próprio autor

Na Figura 6.4, exibimos a tela com todos os itens já cadastrados referente ao requisito REQ-001 - Manter cadastro de clientes.

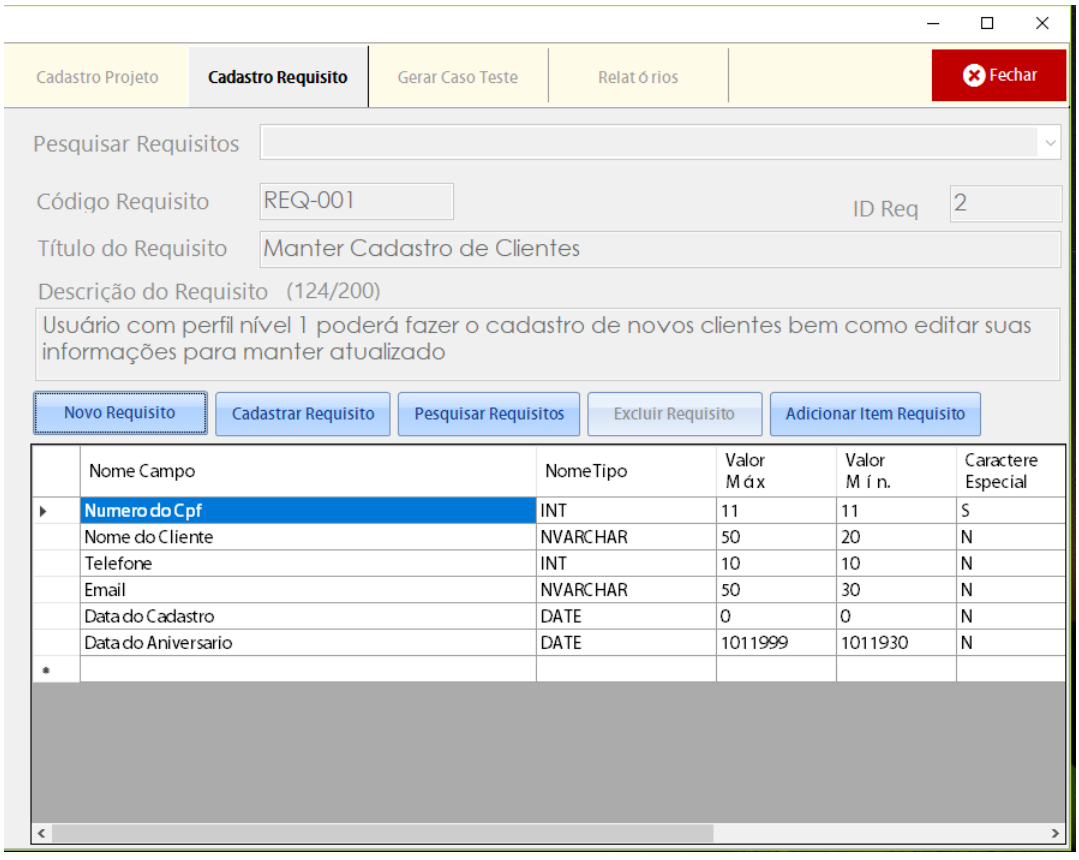


Figura 6.4: Figura Tela de Requisitos com todos os itens cadastrados
Fonte: próprio autor

Na Figura 6.5, temos a funcionalidade para **Gerar Caso de Teste**. Podemos selecionar um requisito na listagem, que neste caso foi o Req-002 e clicar no botão **Gerar por**

Requisito.

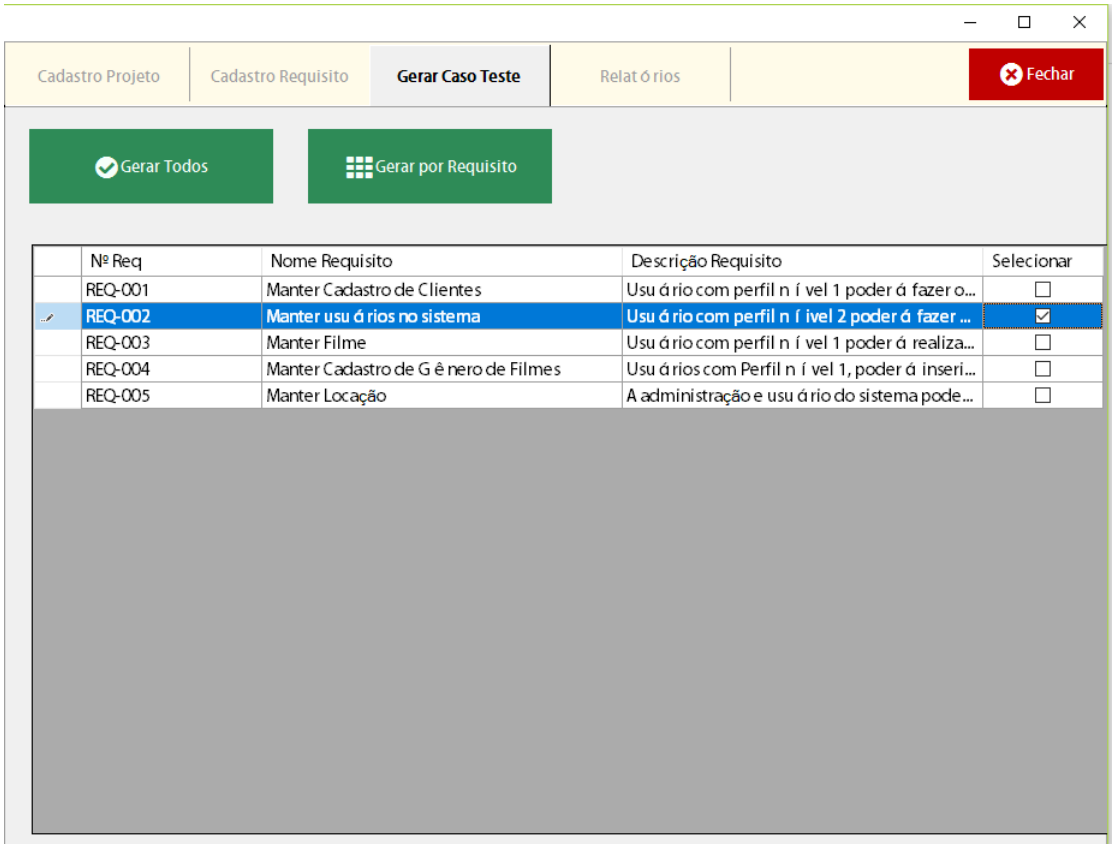


Figura 6.5: Figura Tela de Gerar Caso de Teste
Fonte: próprio autor

A Figura 6.6 demonstra a funcionalidade para **Imprimir** o caso de teste. Podemos selecionar um requisito na listagem que neste caso foi o Req-002 e fazer a impressão clicando no botão com a legenda **Imprimir** para visualizar o relatório.

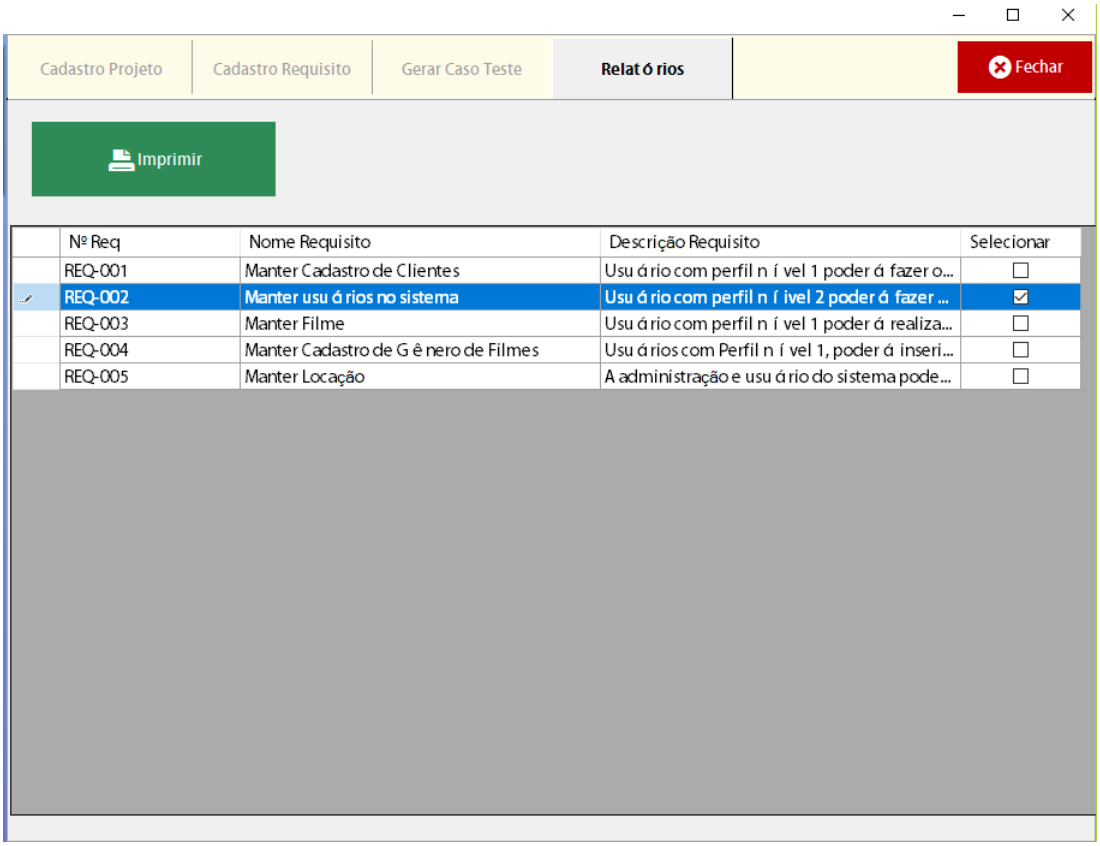
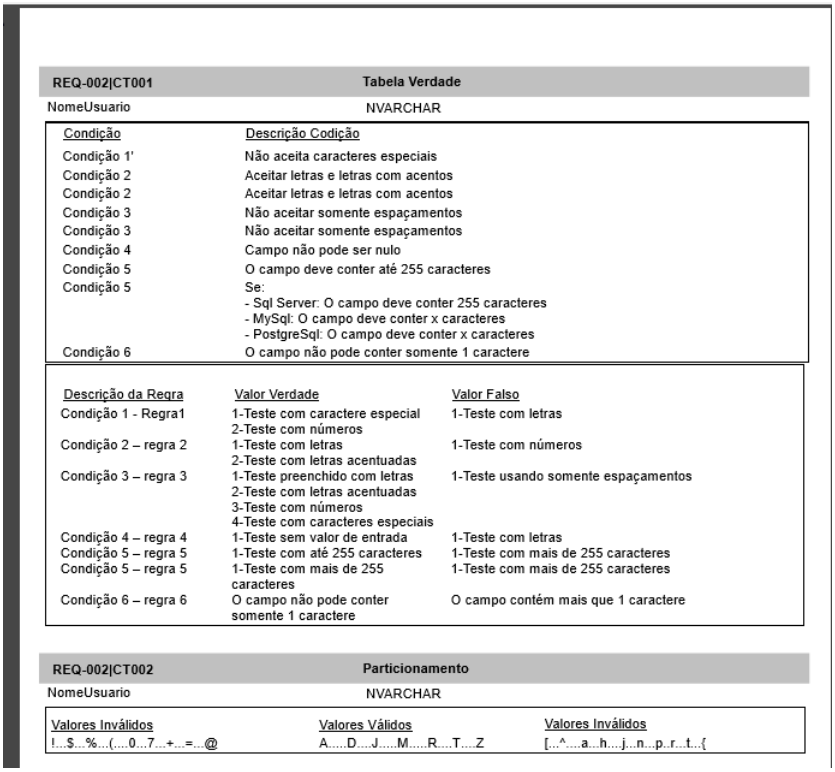


Figura 6.6: Figura Tela de Imprimir Caso de Teste
Fonte: próprio autor

Logo após clicar no botão imprimir citado na Figura 6.6 o relatório será apresentado como demonstra a Figura 6.7.



REQ-002|CT001	Tabela Verdade
NomeUsuario	NVARCHAR

Condição	Descrição Codição
Condição 1'	Não aceita caracteres especiais
Condição 2	Aceitar letras e letras com acentos
Condição 2	Aceitar letras e letras com acentos
Condição 3	Não aceitar somente espaçamentos
Condição 3	Não aceitar somente espaçamentos
Condição 4	Campo não pode ser nulo
Condição 5	O campo deve conter até 255 caracteres
Condição 5	Se: - Sql Server: O campo deve conter 255 caracteres - MySql: O campo deve conter x caracteres - PostgreSql: O campo deve conter x caracteres
Condição 6	O campo não pode conter somente 1 caractere

Descrição da Regra	Valor Verdade	Valor Falso
Condição 1 - Regra1	1-Teste com caractere especial 2-Teste com números	1-Teste com letras
Condição 2 – regra 2	1-Teste com letras 2-Teste com letras acentuadas	1-Teste com números
Condição 3 – regra 3	1-Teste preenchido com letras 2-Teste com letras acentuadas 3-Teste com números 4-Teste com caracteres especiais	1-Teste usando somente espaçamentos
Condição 4 – regra 4	1-Teste sem valor de entrada	1-Teste com letras
Condição 5 – regra 5	1-Teste com até 255 caracteres	1-Teste com mais de 255 caracteres
Condição 5 – regra 5	1-Teste com mais de 255 caracteres	1-Teste com mais de 255 caracteres
Condição 6 – regra 6	O campo não pode conter somente 1 caractere	O campo contém mais que 1 caractere

REQ-002|CT002	Particionamento
NomeUsuario	NVARCHAR

Valores Inválidos	Valores Válidos	Valores Inválidos
!...$...%...(....0...7...+...=...@	A.....D.....J.....M.....R.....T.....Z	[...^.....a...h....j...n...p..r...t...{

Figura 6.7: Figura Tela do Relatório do Caso de Teste gerado
Fonte: próprio autor

6.3 Revisão do Capítulo

Neste capítulo exibiu-se o resultado da ferramenta para a geração automática de casos de teste em um estudo de cado de um software de gestão de uma locadora. Foi utilizado um formulário com alguns campos para demonstrar as funcionalidades proposta. Ao final é exibido um relatório descrevendo os casos de testes criados pela ferramenta.

Capítulo 7

Conclusão

Não há como desvincular os testes de software quando se fala em qualidade. É de extrema importância para os profissionais de TI garantir a eficácia na produção de software, no que rege um planejamento com excelência.

Após a realização das pesquisas para a construção deste trabalho, constatamos que o processo de teste deve ser vinculado no início do desenvolvimento do projeto. O ciclo de teste prossegue mesmo depois da implantação do software, durante os processos de manutenções evolutivas e corretivas.

Utilizando-se de tecnologias de engenharia de software é possível chegar a um produto mensurável e de extrema performance, elevando a sua qualidade final. Olhando do ponto do vista do programador, ter em mãos uma ferramenta que possa auxiliar nos testes, permite um controle adequado a cada fase do processo de desenvolvimento do projeto.

A demanda de produtos gratuitos no mercado atual, que auxiliem no processo de geração de caso de teste, podem contribuir para o aperfeiçoamento do processo de desenvolvimento de software. Este trabalho proporcionou a construção de uma ferramenta cuja a sua principal função é gerar de forma automática documentos para Casos de Testes Funcionais. Utilizou-se as técnicas de Análise do Valor Limite, Particionamento por Equivalência e Tabela Verdade.

Consideramos que os objetivos propostos neste trabalho foram atingidos. Porém, melhorias futuras podem ser incorporadas ao projeto. Como, por exemplo, a adição de novas técnicas de testes funcionais ou até mesmo expandir a ferramenta para testes de caixa branca.

Referências Bibliográficas

actiWATE. actiwate. `https://www.actimind.com/actiwate.html/`, Acesso em 26 set 2017, 16:00hs.

BadBoy. Badboy. `http://www.badboy.com.au//`, Acesso em 26 set 2017, 16:00hs.

GNU. General public licence. `http://www.fsf.org/licenses//`, Acesso em 26 set 2017, 16:00hs.

IEEE-829 (2008). Documentação para padrões de teste de software.

INMETRO. Informação ao consumidor. `http://www.inmetro.gov.br/consumidor/`, Acesso em 01 set 2016, 16:00hs.

KOSCIANSKI, A. (2007). *Qualidade em Software*. Novatec, São Paulo.

MALDONADO, e. C. (2007). *Introdução ao Teste de Software*. Elsevier, Rio de Janeiro.

MECENAS, I. and OLIVEIRA, V. d. (2005). *Qualidade em Software*. Alta Books, Rio de Janeiro.

MOLINARI, L. (2008). *Testes Funcionais de Software*. Alta Books, Florianópolis.

MYERS, G. J. (2012). *The Art of Software Testing 3rd ed.* Parson, New Jersey.

MYSQL (2016). Documentação. informação geral. versão 5.7. `http://www.mysql.com`, Acesso em 25 abril 2016, 10:00hs.

PFLEEGER, S. (2004). *Engenharia de software: teoria e prática*. Prentice Hall, São Paulo.

PRESSMAN, R. (2009). *Engenharia de Software - 7.ed.:*. McGraw Hill Brasil, São Paulo.

RIOS, E. (2010). *Documentação o de Teste de Software: Dissecando o padrão IEEE 829, 2.ed*. Imagem Art Studio, Niteroi.

SOMMERVILLE, I. (2007). *Engenharia de software, 8 ed*. Parson, São Paulo.

WEBtest. Webtest. `http://WEBtest.canoo.com/`, Acesso em 26 set 2017, 16:00hs.

Apêndice A

Email do Leonardo Molinari

Assunto: Ok - Assunto: Material sobre Teste
De: Leonardo Molinari <lm7k@yahoo.com.br>
Data: 20/03/2017 17:08
Para: Jose Roberto <j.robertoamaral@gmail.com>

Jose Roberto,

Tudo bom? É um prazer responder...

Não existe uma ferramenta que gera automaticamente. Os casos de testes tem de ser planejados, escritos. Podem reusados, copiados e reajutados. Existem ferramentas de planejamento e gerencia de testes, como o TestLink. Opensource.

Excecao seria uma ferramenta desenvolvida por uma empresa brasileira que de forma semi automatica gera os casos de testes. Ela reconhece o software (apenas a interface) e depois de ajustes gera de forma parcial. Ganhou premio de inovacao. Na minha dissertacao cito a empresa e o.premio de inovacao que ela ganhou. Mas acredito que perdeu espaco de a revolucao movel.

Sugiro voce comprar 3 livros meus, todos da Editora Erica/Saraiva, sem pestanejar:

-Testes de Software. (Fala de tudo um poucoe ja esta na 4a ed.)
-Inovacao e Automacao de Testes de Software (fala de ferramentas de testes, incluindo testes funcionais)
-Testes de aplicacoes mobile (lancado em janeiro de 2017, fala de testes moveis e tambem faz revisao de tudo, incluindo testes moveis funcionais, testes em jogos que sao essencialmente funcionais). Muito atual.

Ainda tem um livro meu da Editora Visual Books: Testes Funcionais de Software. Ja esta esgotado e so encontra em sebo. Ainda vende em sebo via web. Uso um pedaco dele no meu livro de testes aplicacoes mobile.

Conhecimento eh como uma piramide, a pedra de baixo suporta a pedra de cima.

Ainda existe um projeto de ferramenta de planejamente de testes (para criar casos de testes) usando conceitos de IA. A ferramenta aprenderia a planejar testes a partir de outros testes existentes..Nao sei como esta o projeto hoje, mas quando li a 8 anos atras, estava so em termos teoricos e que era de uma faculdade do nordeste. Acho que de recife. Veja: nao sugiro ir por esse caminho porque o projeto era teorico demais o que incluiria algoritmos de aprendizados. Seria IA aplicada a testes em resumo. Estamos engatinhando nesse sentido.

Espero que tenha ajudado.

Abcs

Leonardo Molinari

Enviado do Yahoo Mail no Android

Apêndice B

Código da Geração dos Casos de Testes construído na ferramenta

E:\ProjAutomatize\AppAutomatize\AppAutomatize\frmPrincipal.cs 1

```
366         private void btnGerarPorRequisito_Click(object sender, EventArgs e)
367         {
368             //Adicionando para o Plano de Teste para cada requisito;
369             var cnn = new MySqlConnection("Persist Security Info = False; server = localhost; database
                  = automatizedb; uid = root;");
370             cnn.Open();
371             MySqlTransaction transacao = cnn.BeginTransaction(IsolationLevel.Serializable);
372
373
374
375
376             //Identificar quantos requisitos existem com filhos e fazere um for
377             //para cada requisito gerar um plano de teste
378             try
379             {
380                 //Plano de Teste
381                 var plano = new PlanoTeste();
382
383                 plano.Projeto_CodigoProjeto = int.Parse(txtCodigoProjeto.Text);
384                 plano.Escopo = txtDescricaoprojeto.Text;
385                 plano.ItemTeste = txtTituloRequisito.Text;
386                 plano.FuncionalidadeTestar = "Verificar";
387
388
389                 var strQ = string.Format("INSERT INTO PlanoTeste (Projeto_CodigoProjeto, Escopo,
                      ItemTeste, FuncionalidadeTestar) " +
390                     "VALUES ('{0}','{1}','{2}','{3}')",
391                      plano.Projeto_CodigoProjeto, plano.Escopo, plano.ItemTeste,
                           plano.FuncionalidadeTestar);
392
393                 var Comando_pt = new MySqlCommand(strQ, cnn, transacao);
394                 Comando_pt.ExecuteNonQuery();
395                 plano.IdPlanoTeste = Comando_pt.LastInsertedId;
396                 // fim
397
398
399                 //Para cada requisito, percorrer todos os itens que o constitui.
400                 //Inserir no no caso de teste de acordo com o tipo de campo aplicando as técnicas de
                      teste funcional caixa prete
401                 //1 particionamento e equivalencia
402                 //2 tabela verdade
403                 //3 Valor limite.
404
405                 //Caso de Teste
406
407                 //Variavel de controles
408                 int i = 0;
409                 int ReqID = 0;
410                 int count_ct = 1;
411                 int min = 0;
412                 int max = 0;
413
414                 //Percorre o datagrid equanto até o final
415                 while (i <= dtGridViewRequisitos.Rows.Count - 1)
416                 {
417                     //Verifica se a linha selecinada está marcada
418                     if (Convert.ToBoolean(dtGridViewRequisitos.Rows[i].Cells["gselect"].Value))
419                     {
420                         //Recupera o valor selecionado no grid
421                         ReqID = int.Parse(dtGridViewRequisitos.Rows[i].Cells["gid"].Value.ToString());
422
423                         //Recupera os itens do requisito selecionado
424                         var app = new Classes.clComporRequisito();
425                         var bindingList = app.ListarItem(ReqID);
426
427                         int id_Tecnica = 0;
428
```

E:\ProjAutomatize\AppAutomatize\AppAutomatize\frmPrincipal.cs 2

```
429                         foreach (var item in bindingList)
430                         {
431                             var ct = new CasoTeste();
432 
433                             //Qual Classificao pertece o campo?
434                             switch (item.Classificacao)
435                             {
436                                 case "1": //Inteiro
437                                     id_Tecnica = 2;
438                                     //Tecnica Particionamento
439                                     if (item.Particionamento)
440                                     {
441                                         var partcionamento = "Particionamento";
442                                         strQ = string.Format("INSERT INTO casoteste(" +
443                                             "NomeDoCasoTeste, " +
444                                             "CodigoRequisito," +
445                                             "ValorMax," +
446                                             "ValorMin," +
447                                             "ValorCentro," +
448                                             "ValorVerdade," +
449                                             "ValorFalso," +
450                                             "PlanoTeste_idPlanoTeste," +
451                                             "PlanoTeste_Projeto_CodigoProjeto,                          ➥
                            ComporRequisito_IdComporRequisito, TecnicaUsada,IdTabVerdade" +
452                                             ") VALUES                                                   ➥
                            ('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}','{11}')",
453                                             item.ConjReq + "|CT" + String.Format("{0:000}", count_ct),
454                                             item.CodigoRequisito,
455                                             item.ValorMaximo,
456                                             item.ValorMinimo,
457                                             1, 1, 1,
458                                             plano.IdPlanoTeste,
459                                             plano.Projeto_CodigoProjeto, item.IDComporRequisito,        ➥
                            partcionamento, 0
460                                             );
461                                         var Comando_ct = new MySqlCommand(strQ, cnn, transacao);
462                                         Comando_ct.ExecuteNonQuery();
463                                         count_ct++;
464                                     }//fim Particionamento
465 
466                                     //Tcnica de ValorLimite
467                                     if (item.AnaliseValorLimente)
468                                     {
469                                         var analise_valor_limite = "Valor Limite";
470                                         //Gerar 4 caso de teste de min e max comparando os valores
471                                         //para cada caso 2 valores  comparando sempre os valores de min ➥
                            e max
472 
473                                         /*Exemplo min>= and max<=  V
474 
475                                                 min< and max> V
476                                                 min< and max<= F
477                                                 min>= and MAx> F
478 
479                                         Criar uma
480                                          */
481                                         int j = 1;
482                                         while (j <= 4)
483                                         {
484 
485                                             switch (j)
486                                             {
487                                                 case 1: //V
488                                                     min = item.ValorMinimo + 1;
489                                                     max = item.ValorMaximo - 1;
490                                                     break;
491 
```

E:\ProjAutomatize\AppAutomatize\AppAutomatize\frmPrincipal.cs 3

```
492                                         case 2: //V
493                                             min = item.ValorMinimo - 1;
494                                             max = item.ValorMaximo + 1;
495                                             break;
496
497                                         case 3://F
498                                             min = item.ValorMinimo - 1;
499                                             max = item.ValorMaximo - 1;
500
501                                             break;
502
503                                         case 4: //F
504                                             min = item.ValorMinimo + 1;
505                                             max = item.ValorMaximo + 1;
506                                             break;
507                                     }
508
509
510                                     strQ = string.Format("INSERT INTO casoteste(" +
511                                     "NomeDoCasoTeste, " +
512                                     "CodigoRequisito," +
513                                     "ValorMax," +
514                                     "ValorMin," +
515                                     "ValorCentro," +
516                                     "ValorVerdade," +
517                                     "ValorFalso," +
518                                     "PlanoTeste_idPlanoTeste," +
519                                     "PlanoTeste_Projeto_CodigoProjeto,
                            ComporRequisito_IdComporRequisito, TecnicaUsada,IdTabVerdade" +
520                                     ") " +
521                                     "VALUES
                            ('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}','{11}')",
522                                     item.ConjReq + "|CT" + String.Format("{0:000}", count_ct),
523                                     item.CodigoRequisito,
524                                     max,
525                                     min,
526                                     0, 0, 0,
527                                     plano.IdPlanoTeste,
528                                     plano.Projeto_CodigoProjeto, item.IDComporRequisito,
                            analise_valor_limite, 0
529                                     );
530                                     var Comando_ct = new MySqlCommand(strQ, cnn, transacao);
531                                     Comando_ct.ExecuteNonQuery();
532                                     count_ct++;
533                                     j++;
534                                 }
535
536                             }
537                             break;
538
539
540                         case "2": //Caracter
541                             id_Tecnica = 1;
542                             //Tabela Verdade
543                             if (item.TabelaVerdade)
544                             {
545                                 var tabelaverdade = "Tabela Verdade";
546                                 strQ = string.Format("INSERT INTO casoteste(" +
547                                     "NomeDoCasoTeste, " +
548                                     "CodigoRequisito," +
549                                     "ValorMax," +
550                                     "ValorMin," +
551                                     "ValorCentro," +
552                                     "ValorVerdade," +
553                                     "ValorFalso," +
554                                     "PlanoTeste_idPlanoTeste," +
555                                     "PlanoTeste_Projeto_CodigoProjeto,
```

E:\ProjAutomatize\AppAutomatize\AppAutomatize\frmPrincipal.cs 4

```
                                ComporRequisito_IdComporRequisito, TecnicaUsada,IdTabVerdade" +
556                                         ") VALUES
                                ('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}','{11}')",
557                                         item.ConjReq + "|CT" + String.Format("{0:000}", count_ct),
558                                         item.CodigoRequisito,
559                                         item.ValorMaximo,
560                                         item.ValorMinimo,
561                                         0, 0, 0,
562                                         plano.IdPlanoTeste,
563                                         plano.Projeto_CodigoProjeto, item.IDComporRequisito,
                                tabelaverdade,1
564                                         );
565                                     var Comando_ct = new MySqlCommand(strQ, cnn, transacao);
566                                     Comando_ct.ExecuteNonQuery();
567                                     count_ct++;
568                                 }//fim tabela
569
570                                  //Tecnica Particionamento
571                                 if (item.Particionamento)
572                                 {
573                                     var partcionamento = "Particionamento";
574                                     strQ = string.Format("INSERT INTO casoteste(" +
575                                         "NomeDoCasoTeste, " +
576                                         "CodigoRequisito," +
577                                         "ValorMax," +
578                                         "ValorMin," +
579                                         "ValorCentro," +
580                                         "ValorVerdade," +
581                                         "ValorFalso," +
582                                         "PlanoTeste_idPlanoTeste," +
583                                         "PlanoTeste_Projeto_CodigoProjeto,
                                ComporRequisito_IdComporRequisito, TecnicaUsada,IdTabVerdade" +
584                                         ") VALUES
                                ('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}','{11}')",
585                                         item.ConjReq + "|CT" + String.Format("{0:000}", count_ct),
586                                         item.CodigoRequisito,
587                                         item.ValorMaximo,
588                                         item.ValorMinimo,
589                                         1, 1, 1,
590                                         plano.IdPlanoTeste,
591                                         plano.Projeto_CodigoProjeto, item.IDComporRequisito,
                                partcionamento,0
592                                         );
593                                     var Comando_ct = new MySqlCommand(strQ, cnn, transacao);
594                                     Comando_ct.ExecuteNonQuery();
595                                     count_ct++;
596                                 }//fim Particionamento
597                                  //Tcnica de ValorLimite
598                                 if (item.AnaliseValorLimente)
599                                 {
600                                     var analise_valor_limite = "Valor Limite";
601                                     //Gerar 4 caso de teste de min e max comparando os valores
602                                     //para cada caso 2 valores  comparando sempre os valores de min
                                 e max
603
604                                     /*Exemplo min>= and max<=  V
605
606                                               min< and max> V
607                                               min< and max<= F
608                                               min>= and MAx> F
609
610                                     Criar uma
611                                      */
612                                     int j = 1;
613                                     while (j <= 4)
614                                     {
615
```

E:\ProjAutomatize\AppAutomatize\AppAutomatize\frmPrincipal.cs 5

```
616                                         switch (j)
617                                         {
618                                             case 1: //V
619                                                 min = item.ValorMinimo + 1;
620                                                 max = item.ValorMaximo - 1;
621                                                 break;
622
623                                             case 2: //V
624                                                 min = item.ValorMinimo - 1;
625                                                 max = item.ValorMaximo + 1;
626                                                 break;
627
628                                             case 3://F
629                                                 min = item.ValorMinimo - 1;
630                                                 max = item.ValorMaximo - 1;
631
632                                                 break;
633
634                                             case 4: //F
635                                                 min = item.ValorMinimo + 1;
636                                                 max = item.ValorMaximo + 1;
637                                                 break;
638                                         }
639
640
641                                         strQ = string.Format("INSERT INTO casoteste(" +
642                                         "NomeDoCasoTeste, " +
643                                         "CodigoRequisito," +
644                                         "ValorMax," +
645                                         "ValorMin," +
646                                         "ValorCentro," +
647                                         "ValorVerdade," +
648                                         "ValorFalso," +
649                                         "PlanoTeste_idPlanoTeste," +
650                                         "PlanoTeste_Projeto_CodigoProjeto,
                                    ComporRequisito_IdComporRequisito, TecnicaUsada,IdTabVerdade" +
651                                         ") " +
652                                         "VALUES
                                    ('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}','{11}')",
653                                         item.ConjReq + "|CT" + String.Format("{0:000}", count_ct),
654                                         item.CodigoRequisito,
655                                         max,
656                                         min,
657                                         0, 0, 0,
658                                         plano.IdPlanoTeste,
659                                         plano.Projeto_CodigoProjeto, item.IDComporRequisito,
                                    analise_valor_limite,0
660                                         );
661                                         var Comando_ct = new MySqlCommand(strQ, cnn, transacao);
662                                         Comando_ct.ExecuteNonQuery();
663                                         count_ct++;
664                                         j++;
665                                     }
666
667                                 }
668
669
670                                 break;
671
672                             case "3": //Data
673                                 id_Tecnica = 3;
674                                 //Tabela Verdade codigo 2
675                                 if (item.TabelaVerdade)
676                                 {
677                                     var tabelaverdade = "Tabela Verdade";
678                                     strQ = string.Format("INSERT INTO casoteste(" +
679                                         "NomeDoCasoTeste, " +
```

E:\ProjAutomatize\AppAutomatize\AppAutomatize\frmPrincipal.cs 6

```
                                        "CodigoRequisito," +
                                        "ValorMax," +
                                        "ValorMin," +
                                        "ValorCentro," +
                                        "ValorVerdade," +
                                        "ValorFalso," +
                                        "PlanoTeste_idPlanoTeste," +
                                        "PlanoTeste_Projeto_CodigoProjeto,
                                ComporRequisito_IdComporRequisito, TecnicaUsada,IdTabVerdade" +
                                        ") " +
                                        "VALUES
                                ('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}','{11}')",
                                        item.ConjReq + "|CT" + String.Format("{0:000}", count_ct),
                                        item.CodigoRequisito,
                                        item.ValorMaximo,
                                        item.ValorMinimo,
                                        0, 0, 0,
                                        plano.IdPlanoTeste,
                                        plano.Projeto_CodigoProjeto, item.IDComporRequisito,
                                tabelaverdade, 3
                                        );
                                    var Comando_ct = new MySqlCommand(strQ, cnn, transacao);
                                    Comando_ct.ExecuteNonQuery();
                                    count_ct++;
                                }//fim tabela
                                break;

                        case "4": //REal
                            id_Tecnica = 2;
                            break;

                        case "5": //Time
                            id_Tecnica = 2;
                            break;

                        case "6": //Boolean
                                  //Tabela Verdade codigo 2
                            if (item.TabelaVerdade)
                            {
                                var tabelaverdade = "Tabela Verdade";
                                strQ = string.Format("INSERT INTO casoteste(" +
                                    "NomeDoCasoTeste, " +
                                    "CodigoRequisito," +
                                    "ValorMax," +
                                    "ValorMin," +
                                    "ValorCentro," +
                                    "ValorVerdade," +
                                    "ValorFalso," +
                                    "PlanoTeste_idPlanoTeste," +
                                    "PlanoTeste_Projeto_CodigoProjeto,
                                ComporRequisito_IdComporRequisito, TecnicaUsada,IdTabVerdade" +
                                    ") " +
                                    "VALUES
                                ('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}','{11}')",
                                    item.ConjReq + "|CT" + String.Format("{0:000}", count_ct),
                                    item.CodigoRequisito,
                                    item.ValorMaximo,
                                    item.ValorMinimo,
                                    0, 0, 0,
                                    plano.IdPlanoTeste,
                                    plano.Projeto_CodigoProjeto, item.IDComporRequisito,
                                tabelaverdade,2
                                    );
                                var Comando_ct = new MySqlCommand(strQ, cnn, transacao);
                                Comando_ct.ExecuteNonQuery();
                                count_ct++;
                            }//fim tabela
```

E:\ProjAutomatize\AppAutomatize\AppAutomatize\frmPrincipal.cs 7

```
741
742
743                                     break;
744                                 }
745
746                             }
747
748                         }
749                         i++;
750                     }
751
752                     transacao.Commit();
753                     MessageBox.Show("Operação concluida com sucesso!", "Mensagem do Sistema",
                          MessageBoxButtons.OK, MessageBoxIcon.Information);
754                     cnn.Close();
755                 }
756                 catch (Exception ex)
757                 {
758
759                     transacao.Rollback();
760                     MessageBox.Show(ex.Message);
761                 }
```

Apêndice C

Relatório do Caso de Teste Gerado no estudo de caso

REQ-002|CT001	**Tabela Verdade**	1
Nome do Usuario	NVARCHAR	

Condição	Descrição Codição
Condição 1'	Não aceita caracteres especiais
Condição 2	Aceitar letras e letras com acentos
Condição 2	Aceitar letras e letras com acentos
Condição 3	Não aceitar somente espaçamentos
Condição 3	Não aceitar somente espaçamentos
Condição 4	Campo não pode ser nulo
Condição 5	O campo deve conter até 255 caracteres
Condição 5	Se: - Sql Server: O campo deve conter 255 caracteres - MySql: O campo deve conter x caracteres - PostgreSql: O campo deve conter x caracteres
Condição 6	O campo não pode conter somente 1 caractere

Descrição da Regra	Valor Verdade	Valor Falso
Condição 1 - Regra1	1-Teste com caractere especial 2-Teste com números	1-Teste com letras
Condição 2 – regra 2	1-Teste com letras 2-Teste com letras acentuadas	1-Teste com números
Condição 3 – regra 3	1-Teste preenchido com letras 2-Teste com letras acentuadas 3-Teste com números 4-Teste com caracteres especiais	1-Teste usando somente espaçamentos
Condição 4 – regra 4	1-Teste sem valor de entrada	1-Teste com letras
Condição 5 – regra 5	1-Teste com até 255 caracteres	1-Teste com mais de 255 caracteres
Condição 5 – regra 5	1-Teste com mais de 255 caracteres	1-Teste com mais de 255 caracteres
Condição 6 – regra 6	O campo não pode conter somente 1 caractere	O campo contém mais que 1 caractere

30	50

REQ-002|CT002	**Particionamento**	0
Nome do Usuario	NVARCHAR	Valor Min de Referência:30 | Valor Max de Referência: 50

Valores Inválidos	Valores Válidos	Valores Inválidos
!...$...%...(....0...7...+...=...@	A.....D....J.....M.....R....T....Z	[...^....a...h....j...n...p..r...t...{

30	50

REQ-002|CT003	**Valor Limite**	0
Nome do Usuario	NVARCHAR	Valor Min de Referência:30 | Valor Max de Referência: 50
31	49	

REQ-002|CT004	**Valor Limite**	0
Nome do Usuario	NVARCHAR	Valor Min de Referência:30 | Valor Max de Referência: 50
29	51	

REQ-002|CT005	**Valor Limite**	0
Nome do Usuario	NVARCHAR	Valor Min de Referência:30 | Valor Max de Referência: 50
29	49	

REQ-002|CT006	**Valor Limite**	0
Nome do Usuario	NVARCHAR	Valor Min de Referência:30 | Valor Max de Referência: 50
31	51	

REQ-002|CT007	**Tabela Verdade**	1
Email	NCHAR	

REQ-002|CT007	Tabela Verdade	1
Email	NCHAR	

Condição	Descrição Codição
Condição 1'	Não aceita caracteres especiais
Condição 2	Aceitar letras e letras com acentos
Condição 2	Aceitar letras e letras com acentos
Condição 3	Não aceitar somente espaçamentos
Condição 3	Não aceitar somente espaçamentos
Condição 4	Campo não pode ser nulo
Condição 5	O campo deve conter até 255 caracteres
Condição 5	Se: - Sql Server: O campo deve conter 255 caracteres - MySql: O campo deve conter x caracteres - PostgreSql: O campo deve conter x caracteres
Condição 6	O campo não pode conter somente 1 caractere

Descrição da Regra	Valor Verdade	Valor Falso
Condição 1 - Regra1	1-Teste com caractere especial 2-Teste com números	1-Teste com letras
Condição 2 – regra 2	1-Teste com letras 2-Teste com letras acentuadas	1-Teste com números
Condição 3 – regra 3	1-Teste preenchido com letras 2-Teste com letras acentuadas 3-Teste com números 4-Teste com caracteres especiais	1-Teste usando somente espaçamentos
Condição 4 – regra 4	1-Teste sem valor de entrada	1-Teste com letras
Condição 5 – regra 5	1-Teste com até 255 caracteres	1-Teste com mais de 255 caracteres
Condição 5 – regra 5	1-Teste com mais de 255 caracteres	1-Teste com mais de 255 caracteres
Condição 6 – regra 6	O campo não pode conter somente 1 caractere	O campo contém mais que 1 caractere

REQ-002|CT008	Particionamento	0
Email	NCHAR	

Valores Inválidos	Valores Válidos	Valores Inválidos
!...$...%...(....0...7...+...=...@	A.....D....J.....M.....R....T....Z	[...^....a...h....j...n...p..r...t...{

REQ-002|CT009	Valor Limite	0
Email	NCHAR	Valor Min de Referência:30 | Valor Max de Referência: 50
31	49	

REQ-002|CT010	Valor Limite	0
Email	NCHAR	Valor Min de Referência:30 | Valor Max de Referência: 50
29	51	

REQ-002|CT011	Valor Limite	0
Email	NCHAR	Valor Min de Referência:30 | Valor Max de Referência: 50
29	49	

REQ-002|CT012	Valor Limite	0
Email	NCHAR	Valor Min de Referência:30 | Valor Max de Referência: 50
31	51	

REQ-002|CT013	Tabela Verdade	1
Email	NCHAR	

REQ-002|CT013	Tabela Verdade
Senha	CHAR(n)

Condição	Descrição Codição
Condição 1'	Não aceita caracteres especiais
Condição 2	Aceitar letras e letras com acentos
Condição 2	Aceitar letras e letras com acentos
Condição 3	Não aceitar somente espaçamentos
Condição 3	Não aceitar somente espaçamentos
Condição 4	Campo não pode ser nulo
Condição 5	O campo deve conter até 255 caracteres
Condição 5	Se: - Sql Server: O campo deve conter 255 caracteres - MySql: O campo deve conter x caracteres - PostgreSql: O campo deve conter x caracteres
Condição 6	O campo não pode conter somente 1 caractere

Descrição da Regra	Valor Verdade	Valor Falso
Condição 1 - Regra1	1-Teste com caractere especial 2-Teste com números	1-Teste com letras
Condição 2 – regra 2	1-Teste com letras 2-Teste com letras acentuadas	1-Teste com números
Condição 3 – regra 3	1-Teste preenchido com letras 2-Teste com letras acentuadas 3-Teste com números 4-Teste com caracteres especiais	1-Teste usando somente espaçamentos
Condição 4 – regra 4	1-Teste sem valor de entrada	1-Teste com letras
Condição 5 – regra 5	1-Teste com até 255 caracteres	1-Teste com mais de 255 caracteres
Condição 5 – regra 5	1-Teste com mais de 255 caracteres	1-Teste com mais de 255 caracteres
Condição 6 – regra 6	O campo não pode conter somente 1 caractere	O campo contém mais que 1 caractere

REQ-002|CT014	Particionamento
Senha	CHAR(n)

Valores Inválidos	Valores Válidos	Valores Inválidos
!...$...%...(....0...7...+...=...@	A.....D....J.....M.....R....T....Z	[...^....a...h....j...n...p..r...t...{

REQ-002|CT015	Valor Limite	
Senha	CHAR(n)	Valor Min de Referência:10 | Valor Max de Referência: 10
11	9	

REQ-002|CT016	Valor Limite	
Senha	CHAR(n)	Valor Min de Referência:10 | Valor Max de Referência: 10
9	11	

REQ-002|CT017	Valor Limite	
Senha	CHAR(n)	Valor Min de Referência:10 | Valor Max de Referência: 10
9	9	

REQ-002|CT018	Valor Limite	
Senha	CHAR(n)	Valor Min de Referência:10 | Valor Max de Referência: 10
11	11	

REQ-002|CT002	Data do Ultimo Acesso	**Particionamento**	DATE

Valores Inválidos	Valores Válidos	Valores Inválidos
!...$...%...(....%...*...+...-..)	0....3....6....9	; : < @ A d j ~

REQ-002|CT019	Data do Ultimo Acesso	**Tabela Verdade**	DATE

Condição	Descrição Codição
Condição 1	Aceitar somente números
Condição 2	A data não pode aceitar somente espaçamento
Condição 3	Quantidade de caracteres = max permitida
Condição 4	Dois primeiros caracteres > 0 e <=31
Condição 5	Dois caracteres do meio >0 <=12
Condição 6	Datas válidas igual min. do banco e max. do banco, caso não tenha sido definido valores mínimos e máximos.

Descrição da Regra	Valor Verdade	Valor Falso
Condição 1 – Regra 1	1-Teste com números	1-Testar com qualquer caractere <> 0...9
Condição 2 – Regra 2	1-Teste com números 2-Teste com espaçamento	1-Teste qualquer caractere <>0...9
Condição 3 - Regra3	1-Teste com números 2-Quantidade de caractere = max.	1-Teste com qualquer caractere <> 0..9
Condição 4 - Regra4	1-Teste com números 2-Teste com números > 0 <=31, para representar o dia	1-Teste com qualquer caractere <> 0..9
Condção 5 - Regra 5	1-Teste com números 2-Teste com números >0 e <=12 para representar mês	1-Teste com qualquer caractere <> 0..9
Condição 6 - Regra6 Condção 6 - Regra 6	1-Teste com números 2-Quantidade de caractere = max. 2-Teste com números >0 e <=31 para dia. 3-Teste com números >0 e <=12 para mês	1-Teste com qualquer caractere <> 0..9