



Instituição Evangélica de Novo Hamburgo
Centro de Educação Profissional - CEP

LUÍS PAULO GOMES LAUER

**AUTOMAÇÃO DE TESTE DE SOFTWARE
COM A FERRAMENTA TESTCOMPLETE**

Novo Hamburgo

2015

LUÍS PAULO GOMES LAUER

**AUTOMAÇÃO DE TESTE DE SOFTWARE
COM A FERRAMENTA TESTCOMPLETE**

**Trabalho de Conclusão de Curso, Pré-requisito
para conclusão do curso Técnico em
Informática, fazendo jus ao título conferido pela
INSTITUIÇÃO EVANGÉLICA DE NOVO
HAMBURGO- CENTRO SINODAL DE ENSINO
MÉDIO DE NOVO HAMBURGO - Unidade de
Ensino FUNDAÇÃO EVANGÉLICA**

ORIENTADOR: Prof. Vlademir B. Viana

Novo Hamburgo

2015

RESUMO

Na era da tecnologia, o mercado de software está cada vez mais exigente e a qualidade da aplicação é um quesito importante para a aquisição de um novo software. O teste de software, juntamente com a automação de teste, torna-se muito importante para que a qualidade do software alcance um padrão satisfatório, porém muitas empresas ainda não possuem práticas eficazes para aplicar testes em seus softwares ou muitas vezes nem o fazem, deixando somente a cargo do desenvolvedor do sistema. Essa falta de testes acarreta em mau funcionamento do software e pode aumentar significativamente o custo do desenvolvimento do mesmo, devido ao retrabalho em corrigir as falhas depois que o mesmo já tenha sido entregue ao cliente. Este trabalho apresenta um estudo sobre a aplicação de um projeto de automação de teste de software utilizando a ferramenta *TestComplete*. O projeto foi aplicado no Software ERP GMAX, da empresa Genesys Engenharia de Software LTDA, o qual constantemente sofre atualizações.

Palavras-chave: Testabilidade de software. Automação de teste de software. Ferramenta *TestComplete*.

ABSTRACT

In the age of technology, the software market is increasingly demanding and application quality is an important requirement for the purchase of new software. Software testing, with test automation, it becomes very important for the quality of the software reach a satisfactory standard, but many companies still do not have effective practices to apply testing in their software or often do not do so, leaving only in charge of the system developer. This leads to lack of testing software malfunction and can significantly increase the cost of developing the same, due to the rework to correct the flaws after it has been delivered to the customer. This paper presents a study on the implementation of a software test automation project using TestComplete tool. The project was implemented in ERP Software GMAX, the company Genesys Software Engineering LTD, which constantly suffers updates.

Keywords: Testability of Software. Software Test Automation. TestComplete Tools.

LISTA DE ILUSTRAÇÕES

Figura 1 - Fases de Teste de Software	13
Figura 2 - Modelo V descrevendo o paralelismo entre as atividades de desenvolvimento e teste de software	15
Figura 3 - TestComplete Método ADO.CreateADOQuery	18
Figura 4 - TestComplete Método Process Object.....	19
Figura 5 - TestComplete Hierarquia do Método Process Object.	20
Figura 6 - TestComplete Método Log.Messag	21
Figura 7 - TestComplete Método AddPicture.	22
Figura 8 - TestComplete Método Compare.	23
Figura 9 - Diagrama dos casos de uso.....	26
Figura 10 - TestComplete New Project Suite	27
Figura 11 - TestComplete Create Project Suite.....	28
Figura 12 - TestComplete Create New Project.....	28
Figura 13 - TestComplete Select Project Items	29
Figura 14 - TestComplete Alter Name Script.....	29
Figura 15 - TestComplete Script Main.....	30
Figura 16 - TestComplete novo script	30
Figura 17 - Tabela ADM_ROTINA.....	31
Figura 18 - Procedure PesquisaBanco.....	32
Figura 19 - Procedure AbreTelaRotina.....	33
Figura 20 - Formulário Acesso a Rotinas.	34
Figura 21 - Formulário TAB.2. Empresas	34
Figura 22 - TestComplete start recording.	35
Figura 23 - TestComplete Start Recording ações do usuário.....	36
Figura 24 - Script de teste IncluiCadastro criado na linguagem DelphiScript.	37
Figura 25 - Redução do script de teste IncluiCadastro.....	38
Figura 26 - Procedure TestaInclusaoCadastroBanco.....	39
Figura 27 - Select da tabela TAB000002 - TestaInclusaoCadastroBanco.	40
Figura 28 - Condição - Procedure TestaModificacaoCadastroBanco.....	41
Figura 29 - Condição - Procedure TestaExclusaoCadastroBanco.	41
Figura 30 - Select da tabela TAB000002 - TestaExclusaoCadastroBanco.	41

Figura 31 - Formulário ATB.3.2.Remessas bancárias.....	42
Figura 32 - Script - procedure TestaRelatorio_ATB_3_2.	43
Figura 33 - Object Properties - CheckBox Imprime dados cadastrais.	44
Figura 34 - Método wState - CheckBox Imprime dados cadastrais.....	45
Figura 35 - Dispositivo de saída do relatório	45
Figura 36 - Salvar Imagem no Stores.....	46
Figura 37 - Compara Imagens.....	47
Figura 38 - Execução do Project Suite	48
Figura 39 - Resultado da execução do Project Suite	48
Figura 40 - Erro do teste Project TAB_2	49
Figura 41 - Log de erro das imagens diferentes.....	50
Figura 42 - Resultado das imagens diferentes	50

SUMÁRIO

1	INTRODUÇÃO	9
1.1	CONTEXTO	9
1.2	OBJETIVOS	9
1.2.1	<i>Objetivos específicos</i>	10
1.3	JUSTIFICATIVA	10
1.4	ESTRUTURA DO TRABALHO	10
2	FUNDAMENTAÇÃO TEÓRICA	11
2.1	TESTABILIDADE DE SOFTWARE	11
2.2	AUTOMAÇÃO DE TESTE DE SOFTWARE	16
2.3	FERRAMENTA TESTCOMPLETE	17
2.4	MÉTODOS DO TESTCOMPLETE	18
2.4.1	<i>ADO.CreateADOQuery</i>	18
2.4.2	<i>BuiltIn.ShowMessage</i>	18
2.4.3	<i>Stop Method</i>	19
2.4.4	<i>Process Object</i>	19
2.4.5	<i>Delay Method</i>	20
2.4.6	<i>VisibleOnScreen Property</i>	20
2.4.7	<i>ActiveWindow (Desktop Objects)</i>	20
2.4.8	<i>Log Object</i>	20
2.4.9	<i>BuiltIn.Delay</i>	22
2.4.10	<i>AddPicture Method</i>	22
2.4.11	<i>Compare Method</i>	23
2.5	LINGUAGEM SQL	23
3	METODOLOGIA	24
3.1	CARACTERIZAÇÃO DO PROJETO	24
3.2	PESQUISAS E TAREFAS	24
3.3	TECNOLOGIAS UTILIZADAS	25
4	DESENVOLVIMENTO	26

4.1	CARACTERIZAÇÃO DA EMPRESA.....	26
4.2	DIAGRAMA DE CASOS DE USO	26
4.3	PROJETO DE TESTE COM TESTCOMPLETE	27
5	RESULTADOS	48
6	CONCLUSÃO	51

1 INTRODUÇÃO

Este trabalho aborda a automação de teste de software aplicada com a ferramenta *TestComplete*. A automação de teste de software condiz em utilizar uma ferramenta para efetuação do teste, tendo em vista a redução do envolvimento humano em atividades manuais repetitivas. O teste de software é a investigação do software a fim de fornecer informações sobre sua qualidade em relação ao contexto em que ele deve operar. Isso inclui o processo de utilizar o produto para encontrar seus defeitos. O teste de software é um processo realizado pelo testador de software, que permeia outros processos da engenharia de software, e que envolve ações que vão do levantamento de requisitos até a execução do teste propriamente dito.

1.1 CONTEXTO

Atualmente, empresas de tecnologia que desenvolvem software que constantemente sofrem novas atualizações, necessitam efetuar testes repetitivos baseados em casos de uso, com o propósito de encontrar possíveis falhas devido a um novo componente integrado ao sistema ou devido a alteração de um componente existente.

Estes testes se efetuados manualmente, retém uma grande quantia de tempo do testador, em muitos casos os resultados não são confiáveis pelo motivo de haver a possibilidade de sofrer alterações de rotinas de teste caso o testador cometa uma falha na efetuação do mesmo.

A automação de teste de software, realizada por ferramentas específicas para esta finalidade, pode detectar muitas falhas que o analista de teste não detectaria ao efetuar o teste manualmente, principalmente em testes de repetição.

1.2 OBJETIVOS

Utilizar um software específico para automação de testes aplicados em software, com intuito de aumentar a qualidade dos testes, reduzindo o tempo de execução e trazendo maior confiabilidade nos resultados dos testes aplicados no software ERP GMAX.

1.2.1 Objetivos específicos

- Buscar conhecimentos sobre teste de software ;
- Buscar conhecimentos nas regras de negócio para utilização da automação de testes;
- Buscar conhecimentos para automação de teste de software;
- Buscar conhecimentos em linguagem SQL;
- Buscar conhecimentos para utilização da ferramenta *TestComplete*;
- Automatizar por completo o processo de teste de software e liberação da versão para atualização do software ERP GMAX.

1.3 JUSTIFICATIVA

Automatizar o processo de teste de um novo software ou nova versão de um software, faz com que o nível de qualidade se eleve e fortaleça o produto como um todo. Possibilita aumentar a qualidade dos resultados de testes, reduzir o tempo de execução dos testes e reduzir o risco de falhas no software.

1.4 ESTRUTURA DO TRABALHO

Inicialmente será apresentada a fundamentação teórica que abordará a testabilidade de software seção(2.1), automação de teste de software seção(2.2), ferramenta *TestComplete* seção (2.3), métodos do *TestComplete* seção(2.4) e linguagem SQL seção(2.5).

Em seguida, será apresentada a metodologia que contém a caracterização do projeto seção (3.1), pesquisas e tarefas seção (3.2) e tecnologias utilizadas seção (3.3).

No desenvolvimento encontra-se a caracterização da empresa seção (4.1), diagrama dos casos de uso seção (4.2) e o projeto de teste com *TestComplete* seção(4.3).

Por fim, os resultados e a conclusão/considerações finais bem como as referências bibliográficas.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta o referencial teórico sobre testabilidade de software, automação de teste de software, utilização de ferramentas específicas para a automação de teste de software, ferramenta (*TestComplete*), métodos da ferramenta *TestComplete* e conceitos da linguagem SQL.

2.1 TESTABILIDADE DE SOFTWARE

"A área de Testes de Software vem se tornando cada vez mais importante para que o desenvolvimento alcance um alto nível de qualidade e confiabilidade." (LOURENÇÃO e OLIVEIRA, 2015). Ainda de acordo com (LOURENÇÃO e OLIVEIRA, 2015):

[...]A qualidade de software é definida pela norma ISO/IEC 9126 (2003), como sendo a capacidade de um produto ou serviço apresentar funcionalidades e características que atendem totalmente as necessidades específicas ou implícitas dos usuários.

De acordo com CHATTERJEE, (2004), "testabilidade é um requisito não funcional importante para os membros da equipe de teste e os usuários que estão envolvidos em testes de aceitação do usuário".

Sendo o usuário que irá efetuar a avaliação final do software, caso apresente algum erro no software, na maioria dos casos os desenvolvedores acabam gastando mais tempo para realizar a correção destes erros, muitas vezes complexos, que poderiam ser identificados em sua fase inicial.

Os erros apresentados no software podem aumentar significativamente o custo do desenvolvimento, apura se que, "Um monte de dinheiro é gasto em suporte e manutenção de um produto após o seu desenvolvimento". (CHATTERJEE, 2004).

O teste de software é indispensável quando o objetivo é reduzir as possibilidades de encontrar falhas no sistema, deixando-o menos suscetível a erros que causam frustração para o usuário. Segundo LOURENÇÃO e OLIVEIRA (2015), "a realização dos testes contribui claramente para o aumento da qualidade do programa e resulta na satisfação do usuário final". Assim como na linha exposta por CHATTERJEE (2004), é importante ressaltar que:

[...]Um produto testável garante a execução completa dos *scripts* de teste. Partindo do princípio de que a boa cobertura de teste é aplicada, a maioria dos defeitos serão descobertos e corrigidos antes do lançamento do produto. Isso assegura que os clientes irão relatar um número mínimo de defeitos.

Entretanto, se os testes não são aplicados de forma correta o resultado é semelhante a não efetuação dos mesmos, de fato, na maioria dos casos os testes devem examinar as diferentes probabilidades e características comportamentais que levam o software a falhar se alguma rotina estiver inadequada. (TANNOURI, 2015).

Para que os testes sejam aplicados de maneira correta e apresentem um bom resultado, é importante que a comunicação e informação exista e seja compartilhada entre todos os envolvidos na criação do software, principalmente a integração com o setor de desenvolvimento. Sendo assim, entende-se que esta tarefa não requer apenas um ótimo profissional de testes, segundo TANNOURI (2015), "é muito difícil para um testador identificar problemas se não houver acesso a informações detalhadas sobre os critérios de teste".

[...]Os testadores precisam saber todas as mudanças que ocorreram na especificação, bem como qualquer problema de código ou processo. É necessário garantir que a equipe de testes sabe quais áreas do produto estão prontas para testar e quais não estão. Isto economiza tempo e frustrações. (TANNOURI, 2015).

Apesar do teste de software colaborar significativamente para a boa funcionalidade do mesmo, este não pode assegurar a total qualidade do software. De acordo com SOMMERVILLE (2007). A atividade de teste é um processo que tem o objetivo de contribuir para a confiabilidade do software, porém não garante que seu comportamento será o esperado em qualquer circunstância ou que estará totalmente livre de falhas durante sua utilização.

Os testes de software devem ser criados através de estratégias e planejados com rotinas de aplicação de acordo com as necessidades e funcionalidades que o software deve atender, seguindo os casos de uso e de atividade do software, definindo um roteiro que indica o caminho a ser seguido. Estratégia de teste de software reúne alguns elementos que são o planejamento dos testes, o projeto dos casos de teste, a execução dos testes e, por fim, a coleta e avaliação de resultados. (PRESSMAN, 2011). Uma maneira de criação de estratégia de testes é baseando-se na probabilidade de riscos, identificando o componente ou módulo que

possivelmente terá o maior risco de erro e que poderia comprometer o funcionamento do software.

Partindo do princípio de que os testes devem ser aplicados seguindo os casos de teste, MYERS (2004), identifica algumas metodologias eficientes de casos de teste, que, se utilizadas em conjunto, podem desenvolver uma estratégia de teste definitivamente rigorosa, tais metodologias são:

- **Teste de Caixa-preta:** Tem o objetivo de descobrir situações em que o software não se comporta de acordo com as suas especificações, sem se preocupar com os seus aspectos estruturais. Deve ser derivada uma série de condições de entradas, sendo elas válidas ou não, para que o programa seja testado tendo em foco seus requisitos funcionais.
- **Teste de Caixa-branca:** Deve ser testado cada caminho no software pelo menos uma vez, utilizando técnicas específicas para executar decisões lógicas e ciclos em seus limites operacionais, assegurando a validade de suas estruturas de dados.

Os testes de software também possuem fases e etapas, SOMMERVILLE (2007), divide a atividade de Teste de Software em duas fases ou estágios fundamentais que são o Teste de Componentes e o Teste de Sistema. Fases estas que podem ser observadas na figura 1.

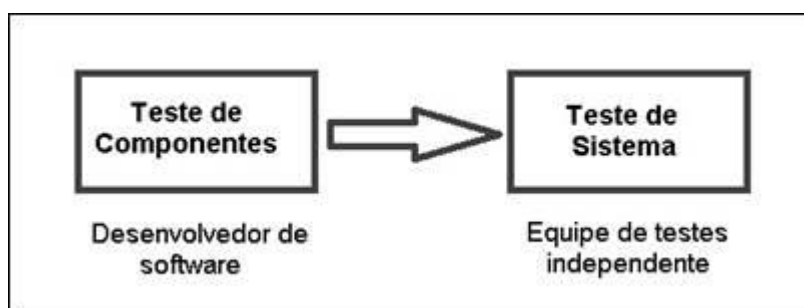


Figura 1 - Fases de Teste de Software
Fonte: (SOMMERVILLE, 2007)

Referente às etapas, os autores MYERS (2004), PRESSMAN (2011) e SOMMERVILLE (2007), citam as seguintes etapas de teste de software:

- **Teste de Unidade:** O foco do teste unitário são os menores blocos ou unidades de um programa, e seu objetivo é facilitar a depuração e apresentar um paralelismo no processo de teste, executando o teste em vários módulos

simultâneos. O teste unitário é considerado como um auxílio para a codificação do software, podendo ser projetado até mesmo antes do código-fonte, sendo que através de informações do projeto, podem ser estabelecidos casos de teste que devem ser ligados a um conjunto de resultados aguardados. Os testes de unidade instituem um escopo restrito no qual se focalizam as estruturas de dados e a lógica interna de processamento.

- **Teste de Integração:** Verifica se, depois de integrados, componentes ou unidades trabalham corretamente em conjunto, com os dados exatos e no tempo esperado através de suas interfaces. Devido à complexidade de algumas interações, a localização dos erros pode se tornar um problema, que com a utilização de uma abordagem incremental, isto é, integrando um conjunto mínimo de componentes e testando-os até que todos componentes estejam devidamente integrados e testados, pode ser resolvido.
- **Teste de Validação:** Logo após o término dos testes de unidade e integração se inicia o teste de validação que, através de um conjunto de testes, deve demonstrar a conformidade com os requisitos e que suas características e especificações estejam de acordo com o esperado, tendo como foco as operações que são visíveis ao usuário final.
- **Teste de Sistema:** O software é somente uma das partes de um sistema de computador, sendo que os testes de sistema agrupam uma série de testes que tem o objetivo de exercitar todas as partes de um sistema, obtendo a garantia de que todos seus elementos funcionam adequadamente. Existem alguns testes específicos que fazem parte do teste de sistema, os mais utilizados e eficientes são os seguintes: Teste de recuperação, Teste de segurança, Teste de esforço, Teste de desempenho e Teste de disponibilização.

Além das etapas de teste citadas acima, também existem as etapas de teste de aceitação e teste de regressão, que de acordo com (NETO 2014 apud ROCHA et al., 2014), podem ser definidas como:

- **Teste de Aceitação:** São realizados geralmente por um restrito grupo de usuários finais do sistema. Esses simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado.

- **Teste de Regressão:** Teste de regressão não corresponde a um nível de teste, mas é uma estratégia importante para redução de “efeitos colaterais”. Consiste em se aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do sistema. Pode ser aplicado em qualquer nível de teste.

Mediante as etapas citadas, NETO (2014), descreve que o planejamento e projeto dos testes devem ocorrer de cima para baixo, ou seja:

1. Inicialmente é planejado o teste de aceitação a partir do documento de requisitos;
2. Após isso é planejado o teste de sistema a partir do projeto de alto nível do software;
3. Em seguida ocorre o planejamento dos testes de integração a partir do projeto detalhado;
4. E por fim, o planejamento dos testes a partir da codificação.

Já a execução dos testes ocorre no sentido inverso como pode ser observado na figura 2:

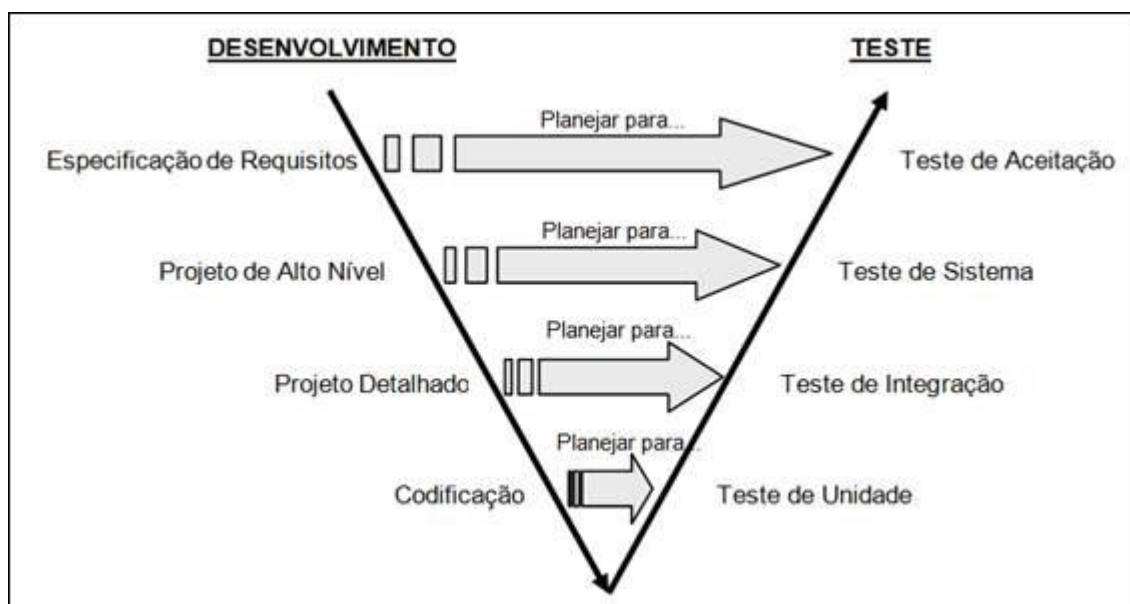


Figura 2 - Modelo V descrevendo o paralelismo entre as atividades de desenvolvimento e teste de software

Fonte: (CRAIG e JASKIEL, 2002)

2.2 AUTOMAÇÃO DE TESTE DE SOFTWARE

Automação de teste de software está relacionada à utilização de ferramentas para execução dos testes, tendo em vista a redução do envolvimento humano em atividades manuais repetitivas e na otimização do tempo de realização do teste, pois a execução de testes automatizados é sempre mais rápida do que os testes manuais e menos suscetível a erros, afinal, a máquina nunca erra. (CAETANO, 2014).

Os testes automatizados interagem com o sistema a ser testado, alguns podem testar um sistema simulando o uso do mesmo por um usuário, geralmente baseando-se na utilização gráfica do sistema, visualização que o usuário irá ter, enquanto que outros aplicam testes em uma parte mais interna do sistema, nas rotinas que formam as regras de negócio. Os tipos de automação são normalmente agrupados de acordo com a forma como os testes automatizados interagem com o sistema. (CAETANO, 2014).

Ainda de acordo com (CAETANO, 2014), em geral temos os seguintes tipos de automação:

- **Baseados na Interface Gráfica:** Nessa abordagem os testes automatizados interagem diretamente com a interface gráfica do sistema simulando um usuário. Normalmente as ações dos usuários são gravadas (*Capture*) por meio da ferramenta. A ferramenta transforma as ações dos usuários em um *script* que pode ser reproduzido (*Playback*) posteriormente.
- **Baseados na Lógica de Negócio:** Nessa abordagem os testes automatizados exercitam as funcionalidades do sistema sem interagir com a interface gráfica. A interface gráfica é apenas uma casca (camada) que tem o objetivo de fornecer um meio para a entrada dos dados e apresentação dos resultados. A camada que abriga a funcionalidade e o comportamento do sistema é a camada de lógica de negócio. Essa abordagem de testes é baseada no entendimento que 80% das falhas estão associados a erros na lógica de negócio, ou seja, existem poucos erros críticos na camada de interface com o usuário. Normalmente é necessário realizar modificações no sistema para torná-lo compatível com essa abordagem. Essas modificações resultam em mecanismos para expor ao mundo exterior as funcionalidades internas do sistema.

Para realizar a automação de teste de software, é importante ter conhecimento em lógica de programação e conhecimento básicos de comandos de linguagem de programação, conhecimentos estes necessários para a criação dos *scripts* de efetuação dos testes, pois na criação dos *scripts* que farão a execução dos testes os mesmos levam padrões da linguagem escolhida.

2.3 FERRAMENTA TESTCOMPLETE

O software *TestComplete* é uma ferramenta desenvolvida pela empresa AutomatedQA. A mesma possibilita automatizar os testes de software, em outras palavras, nada mais é do que um software que irá testar as funcionalidades e características de outro software.

Um dos destaques do *TestComplete* é o seu excelente suporte para a automação de testes de sistemas desenvolvidos em *Delphi*. (CAETANO, 2014). O *TestComplete* não depende de nenhum ambiente de desenvolvimento e pode executar testes automáticos em uma larga variedade de aplicativos como:

- Aplicativos para Windows criados em Visual C++, Visual Basic, Delphi, C++Builder, PowerBuilder, Visual FoxPro e outras ferramentas de desenvolvimento;
- Aplicativos .NET, WPF e Java, web, Flash, Flex e Silverlight;
- Aplicativos para dispositivos móveis, Pocket PCs e smartphones;
- CORBA objects.

A ferramenta *TestComplete* roda em sistema operacional Windows 32bits ou 64bits e pode ser usada para aplicar os diversos tipos de teste, tais como:

- Testes funcionais (interface de usuário);
- Testes em Unidades;
- Testes em regressão;
- Testes White-box;
- Testes por palavra-chave;
- Testes de stress;
- Testes de abrangência;
- Testes Manuais.

2.4 MÉTODOS DO TESTCOMPLETE

A ferramenta *TestComplete* possui métodos que possibilitam criar testes de conexão com o banco de dados e verificação dos dados armazenados em tabelas do banco de dados, a comparação de arquivos e imagens salvas pela própria ferramenta, excluir arquivos armazenados nos diretórios do Windows, gerar log de erros e avisos parando a execução do teste se necessário, permite verificar o estado e as propriedades de componentes do sistema, entre outros métodos que serão apresentados nesta seção.

2.4.1 ADO.CreateADOQuery

O método *ADO.CreateADOQuery*, cria um novo *IAQAADOQuery* objeto e retorna uma referência a ele. Uma vez que o *IAQAADOQuery* objeto é criado, é possível usá-lo para enviar consultas a bancos de dados. (SMARTBEAR, 2015).

O código a seguir na figura 3 ilustra como usar o método *ADO.CreateADOQuery* para consultar dados no bancos de dados.

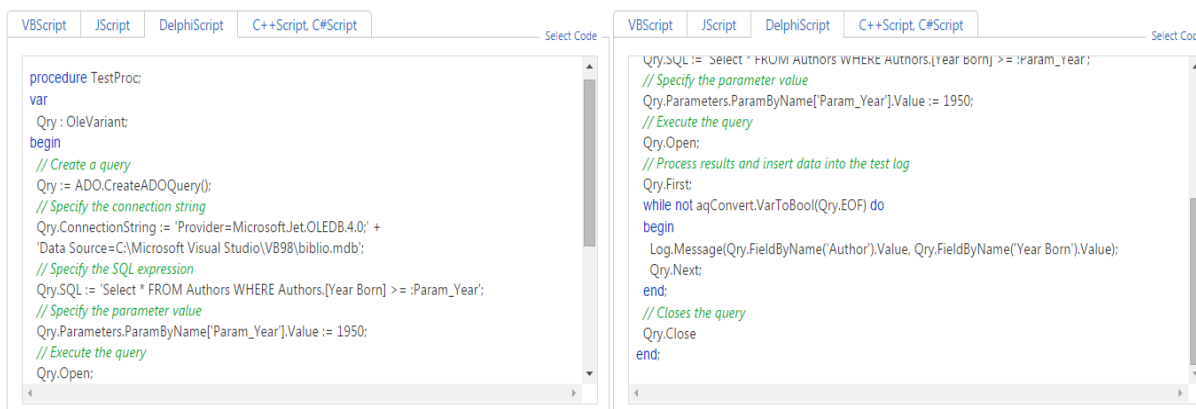


Figura 3 - TestComplete Método ADO.CreateADOQuery
 Fonte: <<https://support.smartbear.com/viewarticle/76243/>>

2.4.2 BuiltIn.ShowMessage

O método *BuiltIn.ShowMessage* exibe uma caixa de mensagem simples com a sequência de caracteres especificada e o botão de *ok*. (SMARTBEAR, 2015).

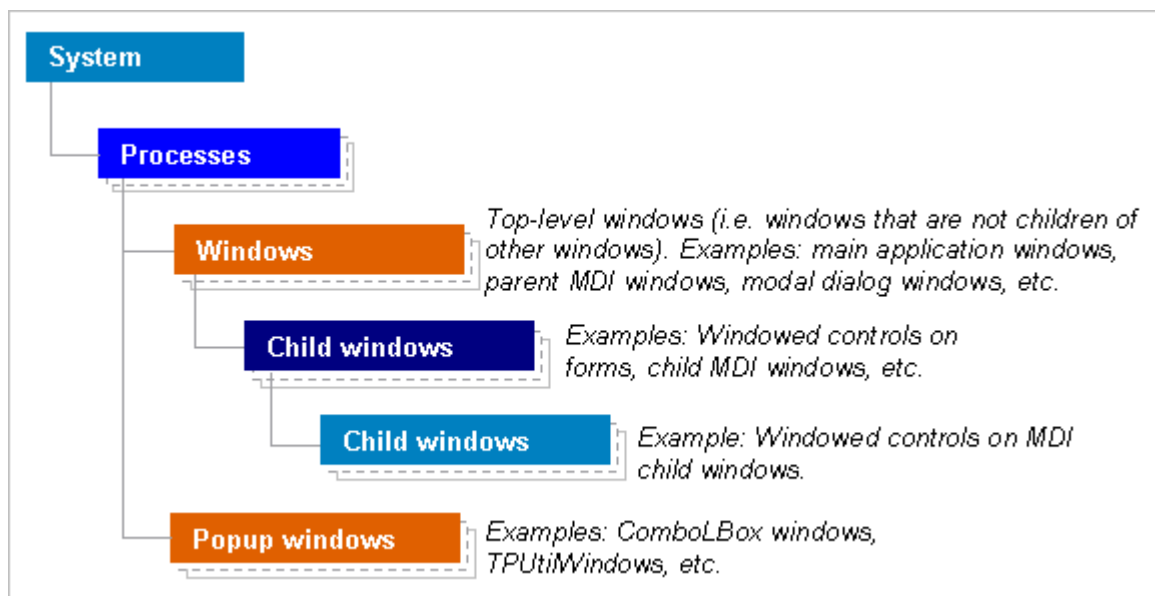


Figura 5 - TestComplete Hierarquia do Método Process Object.
 Fonte: < <https://support.smartbear.com/viewarticle/70041/>>

2.4.5 Delay Method

Método que permite atrasar a execução do *script* por algum período de tempo. Podendo atrasar a execução do teste quando o sistema em teste está realizando algumas operações demoradas. (SMARTBEAR, 2015).

2.4.6 VisibleOnScreen Property

Especifica se o objeto inteiro ou em qualquer das suas partes está atualmente visível na tela. (SMARTBEAR, 2015).

2.4.7 ActiveWindow (Desktop Objects)

Este método pode ser utilizado para obter a janela de nível superior atual (a janela que o usuário está trabalhando atualmente), janela do sistema testado, como uma referência para a janela de objeto. (SMARTBEAR, 2015).

2.4.8 Log Object

TestComplete fornece uma estrutura de *log* de teste que pode armazenar mensagens de texto, imagens, arquivos e links de arquivos gerados a partir de testes. O registro de teste que pode ver através do *log* de teste, pode armazenar

itens como uma lista simples ou tê-los organizados como uma árvore cujos nós são pastas que podem incluir itens de registro de teste e outras pastas.

Sendo possível trabalhar com o *log* de teste a partir de seus *scripts* usando o *log* objeto. Os métodos permitem inserir itens de diferentes tipos em *log* de teste, bem como para criar pastas na log. Na figura 6 é apresentado um exemplo de declaração do método *Log.Message*:

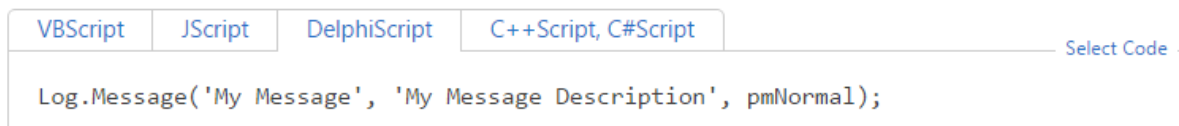


Figura 6 - TestComplete Método Log.Messag
Fonte: < <https://support.smartbear.com/viewarticle/75572/>>

Segue abaixo na tabela 1, a lista dos tipos de log que o *TestComplete* possui:

Nome	Descrição
AppendFolder	Cria uma pasta no log de teste e ativa a pasta de modo que todas as mensagens postadas são enviadas para esta pasta.
Checkpoint	Posta uma mensagem de ponto de verificação para o registro de teste .
CreateFolder	Cria uma pasta no registro de teste. Essa pasta pode conter mensagens de tipos diferentes e subpastas.
CreateIssueFromCurrentLog	Posta o conteúdo do registro de um problema - sistema de rastreamento.
CreateNewAttributes	Cria um novo objeto que define as configurações de fonte e cor a serem aplicadas para testar mensagens e pastas de log.
CreatePictureConfiguration	Cria um novo objeto que irá especificar as definições de formato de imagens postadas no log de teste.
Error	Lança um erro no log de teste.
Event	Posta um evento para o log de teste.
File	Posta um arquivo de log para o teste.
FolderCount	Retorna o número de pastas criadas no log pelo item de teste atual.
Link	Fixa uma referência a um arquivo ou a qualquer outro recurso para o registro de teste .
LockEvents	Desliga o registro de eventos quando UnlockEvents é chamado. O parâmetro Count opcional define o número de eventos mais recentes para se manter em um buffer de segurança. Eles serão adicionados ao log ao postar um erro ou mensagem de aviso.
Message	Posta uma mensagem informativa para o log de teste.
Picture	Posta uma imagem para o registro de teste.

Nome	Descrição
PopLogFolder	Aparece a pasta que está atualmente no topo da pilha de pastas. Isso faz com que a pasta fique no topo da pilha da pasta padrão do registro de teste.
PushLogFolder	Empurra a pasta especificada para o topo da pilha de pasta. Como resultado, por padrão, as mensagens de tipos diferentes serão postadas nesta pasta e novas pastas serão criadas.
SaveResultsAs	Salva os atuais resultados do log de teste em um ou vários arquivos de um formato particular.
SaveToDisk	Gera resultados dos testes intermédios e salva estes na coleção de item de registro.
SetDefaultPictureConfiguration	Especifica as definições de formato de imagens postadas no log de teste.
UnlockEvents	Volta a ativar o registro de eventos.
Warning	Posta um aviso para o log de teste.

Tabela 1 – Lista de Logs *TestCompete*
 Fonte: <<https://support.smartbear.com/viewarticle/69468/>>

2.4.9 BuiltIn.Delay

Atrasa a execução do *script* para o período de tempo especificado, tem a mesma função do *Delay Method*, porém neste pode ser atribuído uma mensagem dentro do parâmetro para que seja apresentada na execução do teste quando chamar este método. (SMARTBEAR, 2015).

2.4.10 AddPicture Method

Método que permite adicionar a imagem especificada para o *Regions* coleção e salvá-la como um item com o nome dado. (SMARTBEAR, 2015). Na figura 7 pode-se observar um exemplo de declaração do método *AddPicture*:

VBScript
JScript
DelphiScript
C++Script, C#Script

Select Code

```

var w : OleVariant;
...
w := Sys.Desktop.ActiveWindow.Picture(20, 20, 50, 50);
// To save the entire window, use
// w := Sys.Desktop.ActiveWindow;
Regions.AddPicture(w, 'MyFile.bmp');
```

Figura 7 - TestComplete Método AddPicture.
 Fonte: <<https://support.smartbear.com/viewarticle/75572/>>

2.4.11 Compare Method

Este método realiza uma comparação de pixel-a-pixel das duas imagens especificadas. Se as imagens forem idênticas, o método retorna *True* e envia uma mensagem informativa para o *log* de teste. Caso contrário, retorna *False* e registra a mensagem apropriada, juntamente com as duas imagens. (SMARTBEAR, 2015). Na figura 8 pode-se observar um exemplo de declaração do método *Compare*:



```

VBScript  JScript  DelphiScript  C++Script, C#Script  Select Code

procedure Test1;
begin
    Pict1 := Utils.Picture;
    Pict2 := Utils.Picture;

    Pict1.LoadFromFile('D:\\Data\\img1.png');
    Pict2.LoadFromFile('D:\\Data\\img2.png');

    // Comparing the images using the Compare method
    if not Pict1.Compare(Pict2) then
    begin
        Log.Picture(Pict1);
        Log.Picture(Pict2);
        Log.Error('The compared images are not identical');
    end;
end;

```

Figura 8 - TestComplete Método Compare.
 Fonte: < <http://support.smartbear.com/viewarticle/72669/>>

2.5 LINGUAGEM SQL

A sigla *SQL*, significa *Structured Query Language*, em português: linguagem de consulta estruturada. Esta linguagem é utilizada em bancos de dados do tipo relacional, e muito importante para a realização da automação do teste de software, pois através do método mencionado anteriormente, método fornecido pela ferramenta *TestComplete*, é possível efetuar a conexão com o banco de dados utilizado juntamente com software testado, banco de dados Oracle do tipo relacional.

"SQL (*Structured Query Language*) é a linguagem padrão universal para manipular bancos de dados relacionais através dos SGBDs. Isso significa que todos os SGBDRs (Sistema de Gerenciamento de Banco de Dados Relacionais) oferecem uma interface para acessar o banco de dados utilizando a linguagem SQL, embora com algumas variações. Logo, saber o que é SQL e como utilizá-la é fundamental para qualquer desenvolvedor de softwares." (FURTADO, 2013).

3 METODOLOGIA

Neste capítulo será apresentada a caracterização do projeto, as pesquisas e tarefas realizadas, bem como as tecnologias utilizadas para o desenvolvimento do projeto de automação de teste de software.

3.1 CARACTERIZAÇÃO DO PROJETO

Este trabalho tem como objetivo automatizar por completo o processo de teste e liberação da versão de atualização do sistema GMAX da empresa Genesys Engenharia de Software LTDA.

Tem-se como público alvo, pessoas que desejam implementar a automação de teste em algum software, de preferência que possuam conhecimento em lógica de programação, pois na criação dos *scripts* de teste são utilizados padrões específicos de programação, *scripts* estes que servem para guiar a execução dos testes.

3.2 PESQUISAS E TAREFAS

As pesquisas bibliográficas foram realizadas em artigos, livros, revistas e sites especializados.

Foi realizado inicialmente um estudo sobre teste de software, analisando quais são os conceitos, metodologias, casos de teste, fases ou estágios de teste e suas etapas.

Na sequência, foi realizada pesquisa sobre automação de teste de software, quais as ferramentas disponíveis no mercado e qual possui suporte para sistemas desenvolvidos na linguagem *Delphi*.

A linguagem *Delphi* foi escolhida para o desenvolvimento dos *scripts* de automação da ferramenta, uma vez que já haviam testes implementados com a ferramenta *TestComplete*, utilizando a linguagem *Delphi*. Ferramenta esta que, segundo (CAETANO, 2014), possui um excelente suporte para a automação de testes de sistemas desenvolvidos em *Delphi*.

Ainda quanto a utilização da ferramenta *TestComplete*, foi dado início a busca de informações sobre as funcionalidades que a ferramenta fornece, e o que poderia ser inovado com a mesma.

Por fim, o estagiário buscou adquirir novos conhecimentos sobre a linguagem SQL, que já havia sido abordada no curso técnico de informática realizado na I.E.N.H (Instituição Evangélica de Novo Hamburgo).

A linguagem SQL foi utilizada para efetuar consultas no banco de dados Oracle, do tipo relacional, que é utilizado para o sistema ERP GMAX, trazendo maior confiabilidade aos testes de cadastro, alteração e exclusão de dados realizados pelo sistema e armazenados no banco de dados.

3.3 TECNOLOGIAS UTILIZADAS

O software testado foi o sistema ERP GMAX da empresa Genesys Engenharia de Software LTDA, "ERPs são softwares que integram todos os dados e processos de uma organização em um único sistema". (GASPAR, 2012). Software este desenvolvido na linguagem *Delphi*.

A ferramenta utilizada para a automação de testes foi o *TestComplete*.

Utilizado o Software da Quest Soft, o Toad para banco de dados Oracle e linguagem SQL para verificação das consultas de banco de dados rodadas nos testes, testes que fazem consultas com o banco de dados verificando a inclusão, alteração ou exclusão de dados.

4 DESENVOLVIMENTO

Neste capítulo descreve-se a caracterização da empresa onde foi realizado o estágio, a integração do teste de software e a automação de teste de software aplicados com a ferramenta *TestComplete* estudadas no referencial teórico, expondo criação dos casos de uso e criação dos projetos com os *scripts* de execução dos testes.

4.1 CARACTERIZAÇÃO DA EMPRESA

Criada em 1991, a Genesys Engenharia de Software tem como missão viabilizar soluções para o sucesso das organizações, por meio da gestão da informação, com tecnologia avançada e inovadora, pessoas valorizadas, e comprometidas, transferindo conhecimento em benefício dos clientes.

4.2 DIAGRAMA DE CASOS DE USO

O sistema ERP GMAX possui muitas Rotinas a serem testadas, atualmente, mais de duzentas. No entanto, priorizou-se dois formulários a serem apresentados neste projeto, pois neles foram aplicados os métodos de teste estudados no referencial teórico. Os casos de uso destes formulários são apresentados na figura 9 do diagrama dos casos de uso.

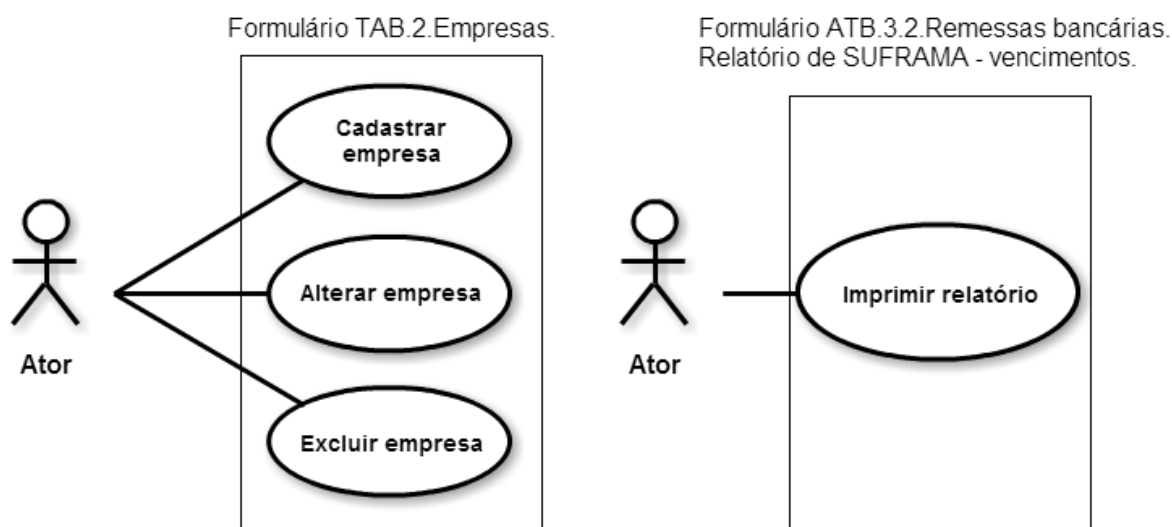


Figura 9 - Diagrama dos casos de uso.
Fonte: Elaborada pelo autor.

4.3 PROJETO DE TESTE COM TESTCOMPLETE

A criação do projeto de teste leva em consideração os casos de uso, as estratégias e rotinas de teste. Teste de regressão foi a etapa de teste utilizada para a criação do projeto de teste, etapa descrita no referencial teórico segundo (NETO apud ROCHA et al., 2014). Nesta seção apresenta-se a criação prática do projeto de teste contendo os *scripts* com as rotinas de teste efetuadas no sistema GMAX.

Na figura 10 é demonstrado como criar um *Project Suite* que possua os demais projetos que poderão ser executados sucessivamente sem pausas.

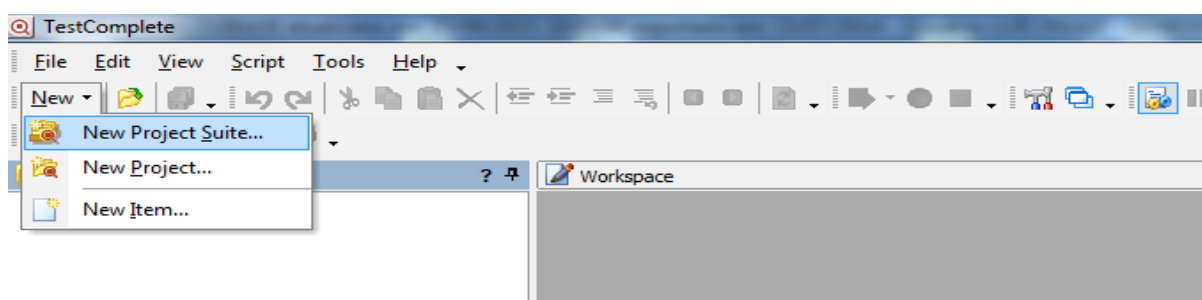


Figura 10 - TestComplete New Project Suite

Fonte: Elaborada pelo autor.

Após clicar em *New Project Suite* será apresentada a janela com campo *Name* e *Location*, para atribuir nome do *Project Suite* e local que será salvo o projeto, o nome foi dado de CALCADOS_BASICOS, pois trata-se de rotinas de teste de clientes calçadistas e por padrão foi mantido o local que o próprio *TestComplete* salva os projetos, conforme ilustrado na figura 11:

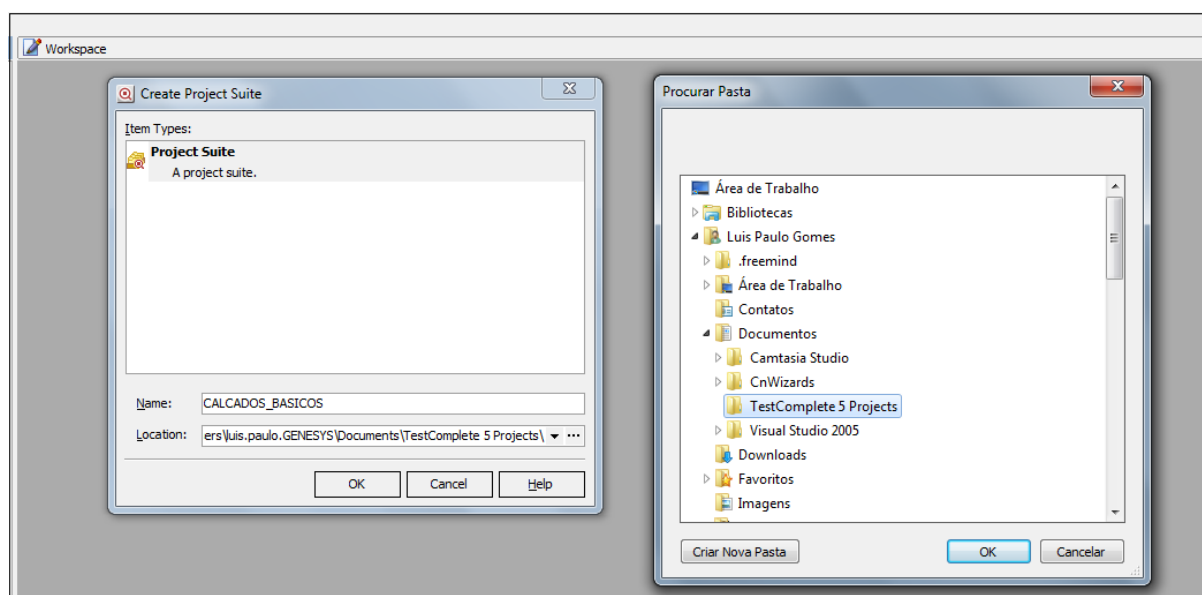


Figura 11 - TestComplete Create Project Suite
Fonte: Elaborada pelo autor.

Clicando no botão ok será criado o *Project Suite*, próximo passo é a criação do projeto de teste que irá conter os *scripts* de teste e demais funcionalidades do *TestComplete*. Novamente em *New* desta vez em *New Project*, irá apresentar a janela onde seleciona o tipo de teste, sendo a *template Unit Testing*, campo *Project name* que foi dado nome de TAB_2, prefixo do formulário testado, e no ultimo campo *Language* é selecionado a linguagem de criação e execução dos *scripts*, neste caso *DelphiScript*. conforme apresenta-se na figura 12:

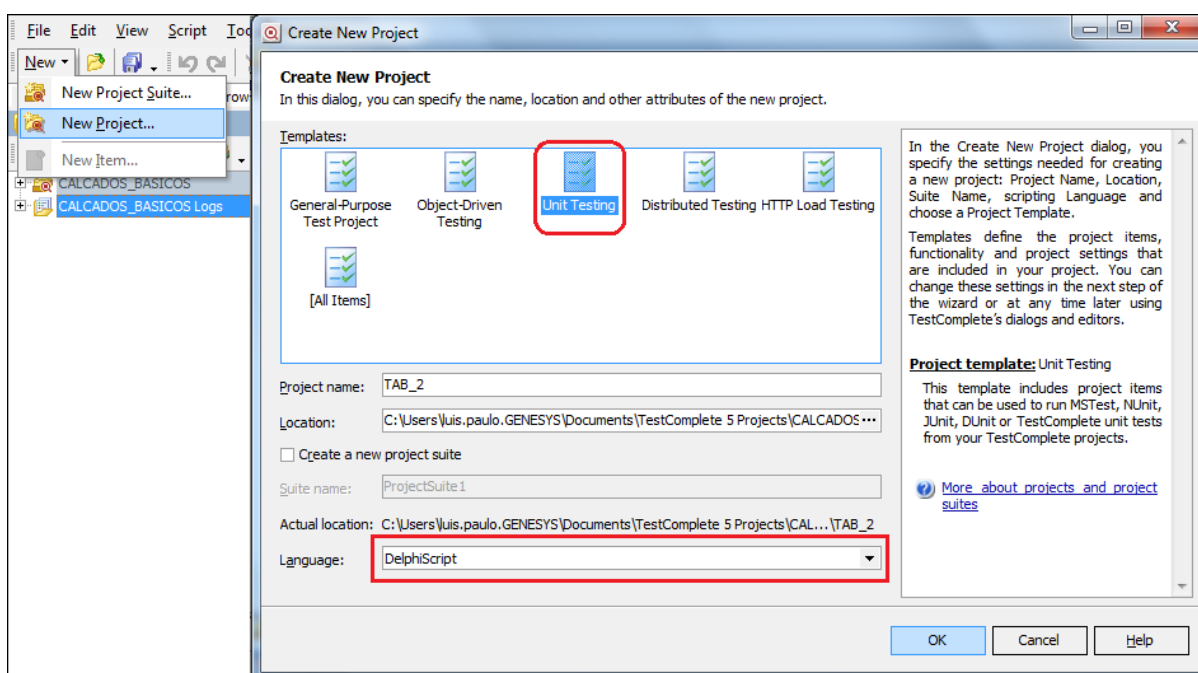


Figura 12 - TestComplete Create New Project
Fonte: Elaborada pelo autor.

Clicando no botão ok será apresentado a próxima janela com os itens do projeto, como *Events*, *Scripts*, *Stores* e etc. Nesta Janela é marcado apenas o item *Stores* e mantém marcado os demais que já estão marcados, depois clicando em *next* até apresentar a janela "*Create New Item*" que refere-se ao *script* de criação e execução dos testes, alterar seu nome para *Main*. Conforme apresenta-se respectivamente nas figuras 13 e 14.

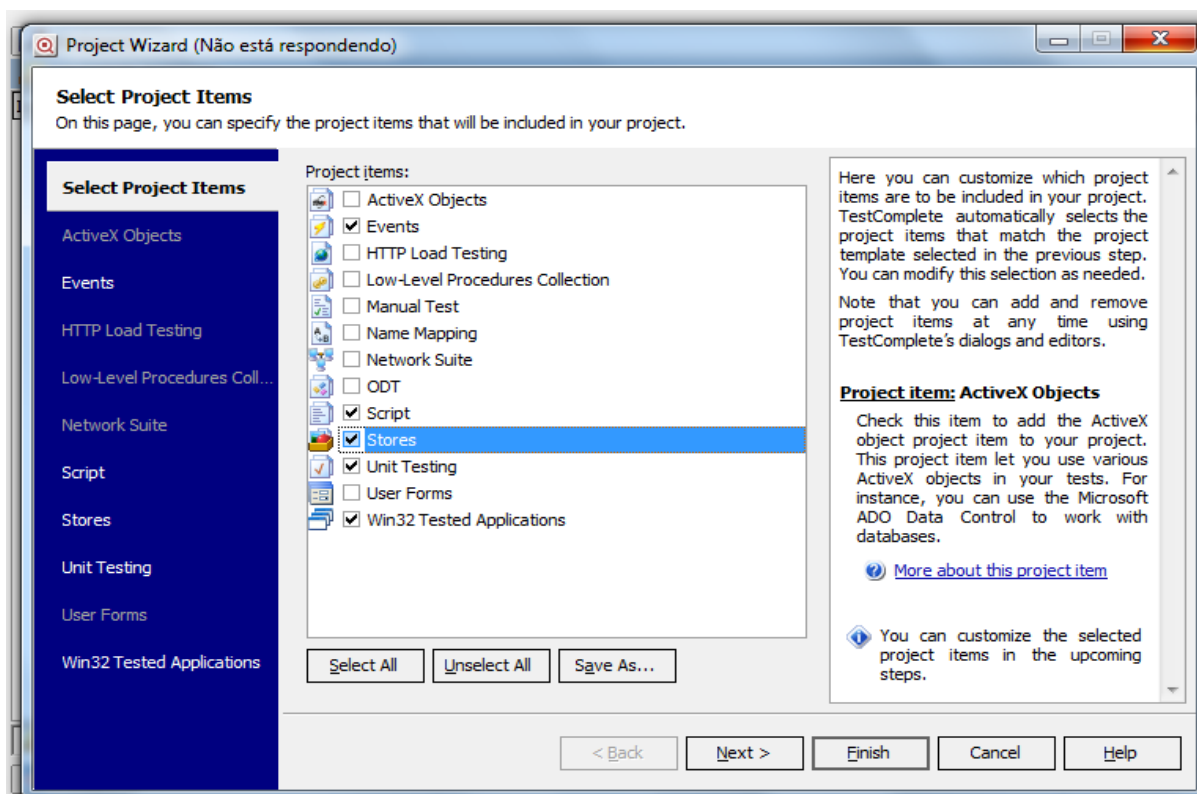


Figura 13 - TestComplete Select Project Items
Fonte: Elaborada pelo autor.

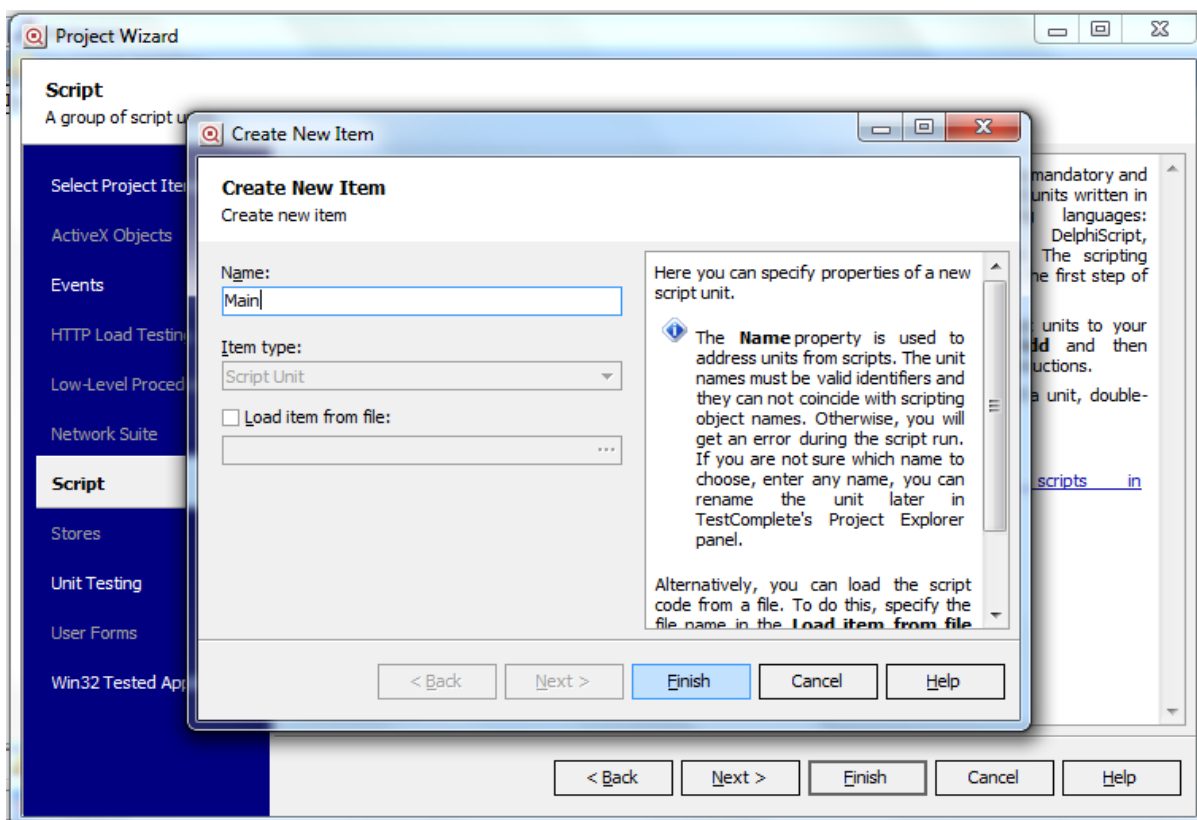


Figura 14 - TestComplete Alter Name Script
Fonte: Elaborada pelo autor.

Após clicar no botão *finish* será criado o *Project* que possuirá os *scripts* de execução dos testes. Veja que já possuí o *script* *Main* que foi nomeado anteriormente, neste *script* irá conter as chamadas das *procedures* dos demais testes que serão executados neste *Project*. *Script* este apresentado na figura 15:

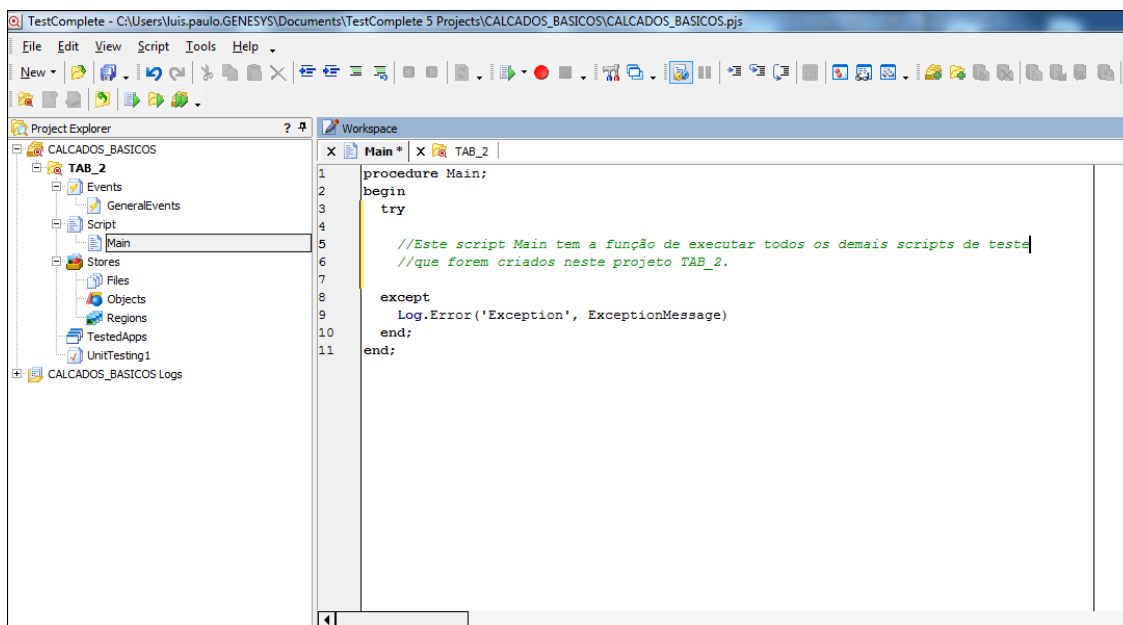


Figura 15 - TestComplete Script Main
Fonte: Elaborada pelo autor.

Foi preciso criar um *script* público, nomeado de *Public_Function*, que possuí *procedures* publicas utilizadas geralmente em todos os *Project* de testes. As *procedure's* principais são a *procedure* **PesquisaBanco** e *procedure* **AbreTelaRotina**. Para adicionar este novo *script* é clicado com o botão direito do mouse em cima do item *Script*, *Add, New Item*, conforme ilustrado na figura 16. Logo apresentará uma janela para inserir o nome do *script*.

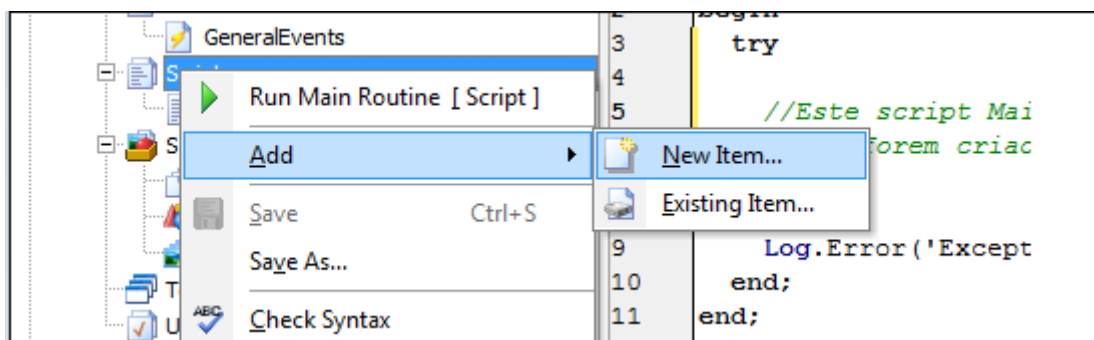
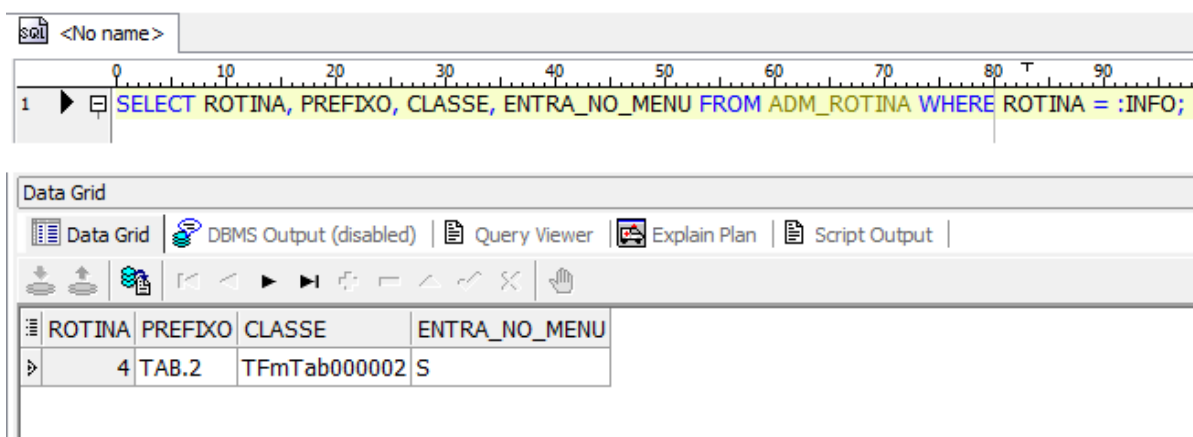


Figura 16 - TestComplete novo script
Fonte: Elaborada pelo autor.

A procedure **PesquisaBanco** tem a finalidade de efetuar a conexão com o banco de dados que possui as tabelas do sistema GMAX, uma tabela em específica, tabela ADM_ROTINA, é utilizada para armazenar as informações do menu que possui o endereço dos formulários do sistema GMAX. Tabela esta onde é verificado a informação dos campos:

- Campo ROTINA: Possui o número da rotina do formulário;
- Campo PREFIXO: Possui o prefixo de nomenclatura do formulário;
- Campo CLASSE: Nome da classe do formulário;
- Campo ENTRA_NO_MENU: Verifica se o formulário esta habilitado no menu, onde "S" quer dizer habilitado e "N" não habilitado.

Campos apresentados na figura 17 após rodar o *select* na tabela ADM_ROTINA, *select* este que contém na *procedure PesquisaBanco*.



The screenshot shows a database query tool interface. At the top, there is a SQL editor with a query: `SELECT ROTINA, PREFIXO, CLASSE, ENTRA_NO_MENU FROM ADM_ROTINA WHERE ROTINA = :INFO;`. Below the editor is a 'Data Grid' window. The Data Grid has a toolbar with icons for refreshing, first, previous, next, last, and other navigation functions. The grid itself displays the results of the query in a table with four columns: ROTINA, PREFIXO, CLASSE, and ENTRA_NO_MENU. The first row of data shows the values 4, TAB.2, TFmTab000002, and S.

ROTINA	PREFIXO	CLASSE	ENTRA_NO_MENU
4	TAB.2	TFmTab000002	S

Figura 17 - Tabela ADM_ROTINA.
Fonte: Elaborada pelo autor.

Segundo (VARELA, 2002), "um procedimento (*procedure*) é semelhante a uma função, sua principal diferença é que um procedimento não tem valor de retorno. Define-se um procedimento obedecendo a seguinte sintaxe:

```
procedure NomeDoProcedimento(Parâmetro1: Tipo; ParâmetroN : Tipo);
  var {declaração de variáveis locais á função quando necessárias}
  begin
    {bloco de instruções}
  end;
```

Abaixo, na figura 18, segue a *procedure* **PesquisaBanco** e a descrição do que esta faz na execução dos testes:

```

37 procedure PesquisaBanco(Info, Pesquisa);
38 var
39   Qry : OleVariant;
40   ClasseRotina;
41 begin
42   try
43     //Cria a Query de conexao com o Banco de Dados.
44     Qry := ADO.CreateADOQuery();
45     //Passa os valores de conexao com o banco de dados.
46     Qry.ConnectionString := 'Provider=PROVEDOR;Data Source=NOME_BASE;User ID=USUARIO;Password=SENHA;';
47
48     //Seleciona a forma de pesquisa do parâmetro 'Pesquisa'.
49     case Pesquisa of
50
51       //Pesquisa a rotina do parâmetro 'Info'.
52       'sOPC' :
53         Qry.SQL := 'SELECT ROTINA, PREFIXO, CLASSE, ENTRA_NO_MENU FROM ADM_ROTINA WHERE ROTINA = :INFO';
54
55     else
56       BuiltIn.ShowMessage('Nenhuma PESQUISA selecionada.');

```

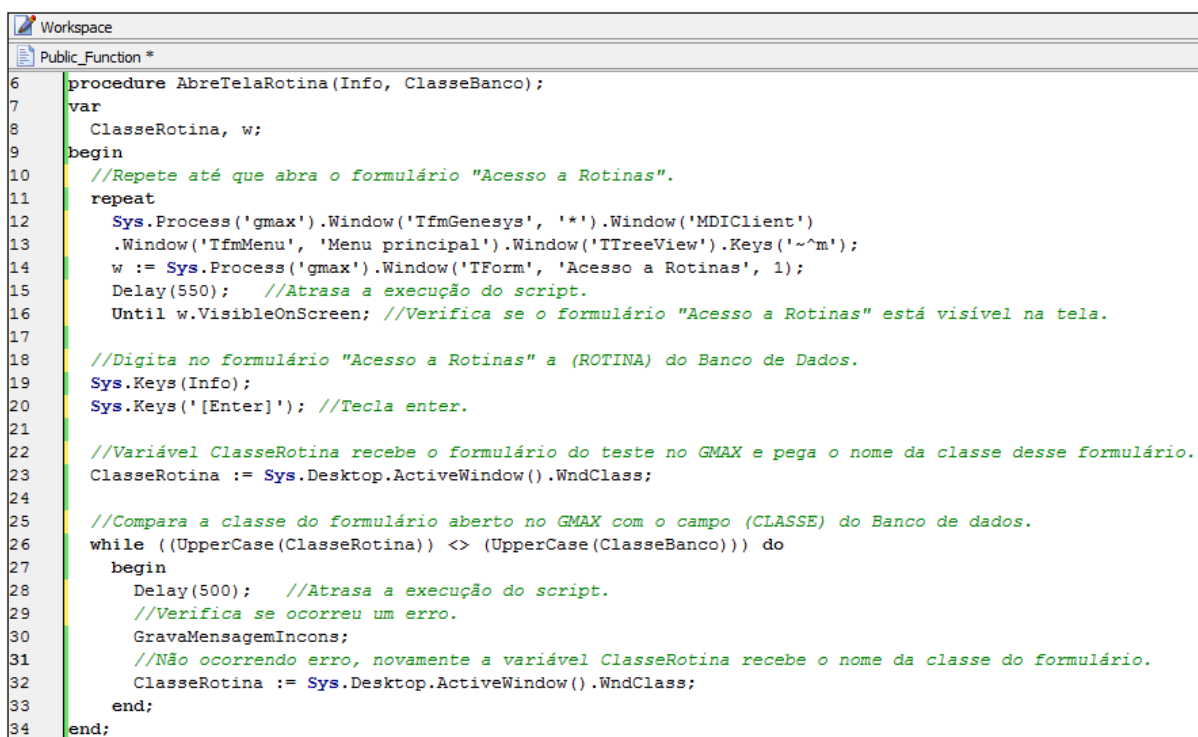
Figura 18 - Procedure PesquisaBanco.
Fonte: Elaborada pelo autor.

Em resumo, esta *procedure* realiza um *select* no banco de dados na tabela ADM_ROTINA, verifica se o formulário está habilitado no menu do GMAX, caso não esteja habilitado, apresenta uma mensagem de erro e para o teste, do contrário pega os valores dos campos ROTINA e CLASSE, passa estes valores no parâmetro

da chamada da *procedure* **AbreTelaRotina**.

A *procedure* **AbreTelaRotina** que é chamada na *procedure* **PesquisaBanco**, tem a finalidade de evidenciar o sistema GMAX e apresentar o formulário Acesso a Rotinas, que foi criado para atalho de abertura dos formulários que serão testados. Pois ao digitar a ROTINA de um formulário na janela de Acesso a Rotinas, este formulário é chamado e apresentado na tela.

Abaixo na figura 19 segue a *procedure* **AbreTelaRotina** e a descrição do que esta faz na execução dos testes:



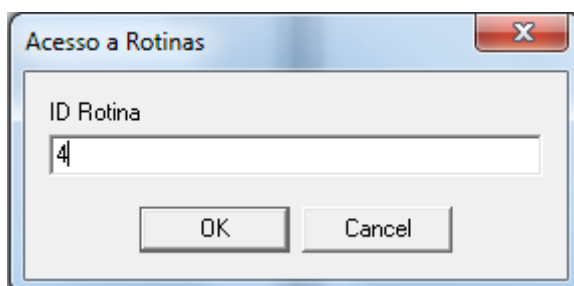
```

6 procedure AbreTelaRotina(Info, ClasseBanco);
7 var
8   ClasseRotina, w;
9 begin
10  //Repete até que abra o formulário "Acesso a Rotinas".
11  repeat
12    Sys.Process('gmax').Window('TfmGenesys', '*').Window('MDIClient')
13    .Window('TfmMenu', 'Menu principal').Window('TTreeView').Keys('~^m');
14    w := Sys.Process('gmax').Window('TForm', 'Acesso a Rotinas', 1);
15    Delay(550); //Atrasa a execução do script.
16    Until w.VisibleOnScreen; //Verifica se o formulário "Acesso a Rotinas" está visível na tela.
17
18    //Digita no formulário "Acesso a Rotinas" a (ROTINA) do Banco de Dados.
19    Sys.Keys(Info);
20    Sys.Keys('[Enter]'); //Tecla enter.
21
22    //Variável ClasseRotina recebe o formulário do teste no GMAX e pega o nome da classe desse formulário.
23    ClasseRotina := Sys.Desktop.ActiveWindow().WndClass;
24
25    //Compara a classe do formulário aberto no GMAX com o campo (CLASSE) do Banco de dados.
26    while ((UpperCase(ClasseRotina)) <> (UpperCase(ClasseBanco))) do
27      begin
28        Delay(500); //Atrasa a execução do script.
29        //Verifica se ocorreu um erro.
30        GravaMensagemIncons;
31        //Não ocorrendo erro, novamente a variável ClasseRotina recebe o nome da classe do formulário.
32        ClasseRotina := Sys.Desktop.ActiveWindow().WndClass;
33      end;
34 end;
  
```

Figura 19 - Procedure AbreTelaRotina.
Fonte: Elaborada pelo autor.

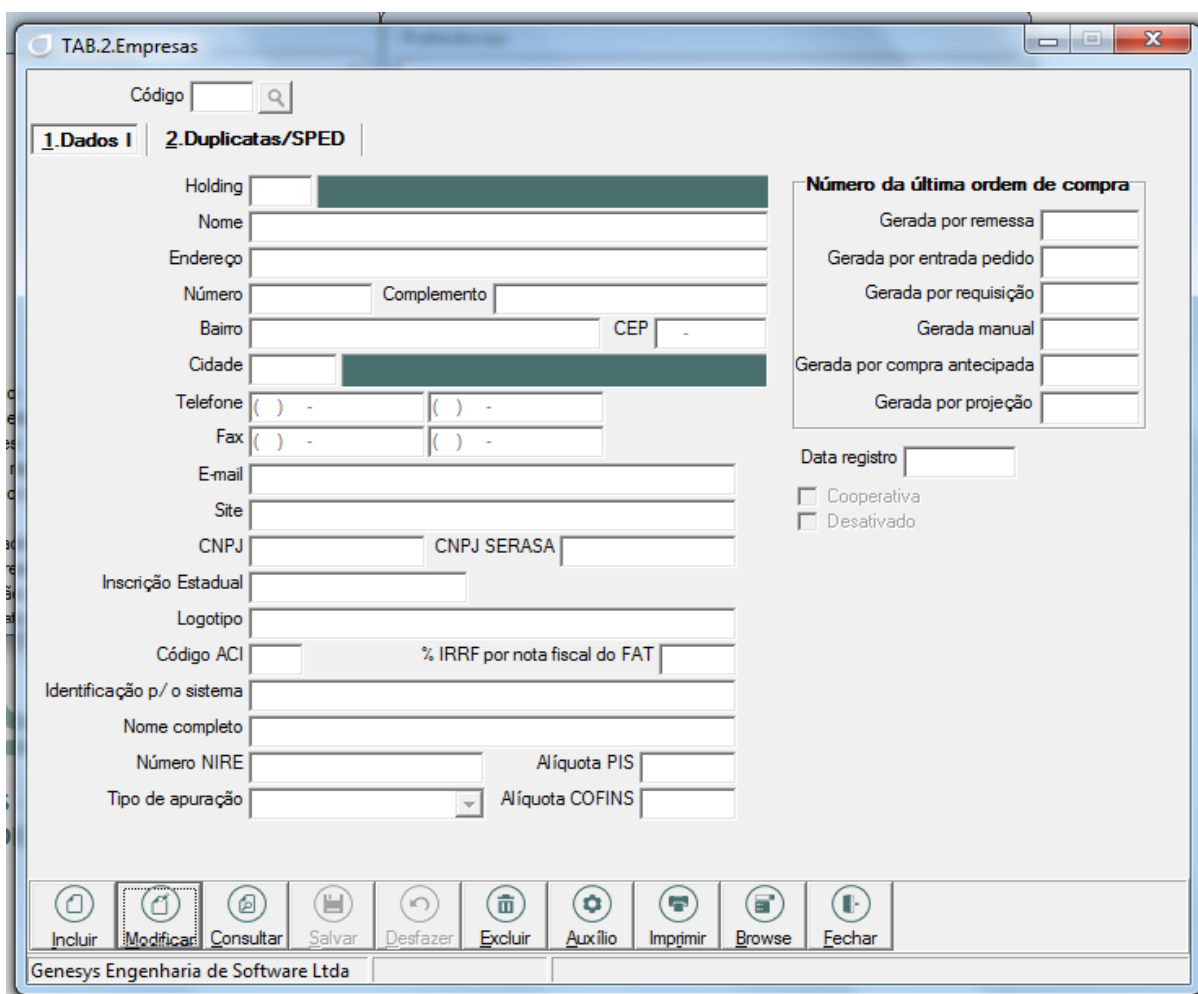
Em resumo, esta *procedure* digita no formulário Acesso a Rotinas, o valor do campo ROTINA da tabela ADM_ROTINA e tecla no *enter*, ao teclar no *enter* irá abrir o formulário da ROTINA, quando efetua a abertura deste formulário pega o nome da classe deste formulário pelo método *Sys.Desktop.ActiveWindow().WndClass*, depois de pegar o nome da classe do formulário, compara com o nome do campo CLASSE da tabela ADM_ROTINA, se os valores forem diferentes chama a *procedure* **GravaMensagemIncons**, que irá verificar se apresentou na tela alguma mensagem de Atenção ou Inconsistência, se não apresentou irá tentar novamente a verificação da classe.

O formulário Acesso a Rotinas que efetua a abertura dos formulários do menu do GMAX, apresenta-se na figura 20 logo abaixo, seguidamente apresenta-se na figura 21 um dos formulários testados, observe que ao digitar a ROTINA de número 4 no formulário Acesso a Rotinas e teclar o *enter* o formulário do teste é aberto, neste caso o formulário com o nome de TAB.2.EMPRESAS.



Formulário "Acesso a Rotinas" com o título "Acesso a Rotinas" e um botão de fechar (X) no canto superior direito. O formulário contém um campo de texto rotulado "ID Rotina" com o valor "4" digitado. Abaixo do campo, há dois botões: "OK" e "Cancel".

Figura 20 - Formulário Acesso a Rotinas.
Fonte: Elaborada pelo autor.



Formulário "TAB.2.EMPRESAS" com o título "TAB.2.EMPRESAS" e botões de minimizar, maximizar e fechar no canto superior direito. O formulário possui uma barra de pesquisa "Código" com um ícone de lupa. Abaixo, há duas abas: "1. Dados" (selecionada) e "2. Duplicatas/SPED".

1. Dados

Holding Nome

Endereço

Número Complemento

Bairro CEP

Cidade

Telefone () - () -

Fax () - () -

E-mail

Site

CNPJ CNPJ SERASA

Inscrição Estadual

Logotipo

Código ACI % IRRF por nota fiscal do FAT

Identificação p/ o sistema

Nome completo

Número NIRE Alíquota PIS

Tipo de apuração Alíquota COFINS

Número da última ordem de compra

Gerada por remessa

Gerada por entrada pedido

Gerada por requisição

Gerada manual

Gerada por compra antecipada

Gerada por projeção

Data registro

☐ Cooperativa

☐ Desativado

Barra de ferramentas: Incluir, Modificar, Consultar, Salvar, Desfazer, Excluir, Auxílio, Imprimir, Browse, Fechar.

Genesys Engenharia de Software Ltda

Figura 21 - Formulário TAB.2.EMPRESAS
Fonte: Elaborada pelo autor.

Tendo estas duas principais *procedure's* públicas criadas, foi feito a criação do teste deste formulário. A criação do teste segue os casos de uso, onde o cliente pode efetuar três tarefas:

- 1.Cadastro (Incluir) uma empresa;
- 2.Alterar (Modificar) uma empresa cadastrada;
- 3.Excluir uma empresa cadastrada.

A criação do *script* da primeira tarefa e demais tarefas pode ser feita através de um recurso do *TestComplete*, a ferramenta de automação fornece um recurso para gravar (*Capture*) as ações do usuário enquanto o usuário estiver usando a aplicação. Essas ações são traduzidas para comandos na linguagem de *script* suportada pela ferramenta de automação, para que então possam ser executadas (*Playback*) posteriormente. (CAETANO, 2014).

Adicionado ao Project TAB_2 um novo *script* de nome Teste_Inclui, depois de adicionado o *script* efetuou-se a gravação do teste, conforme ilustrado na figura 22:

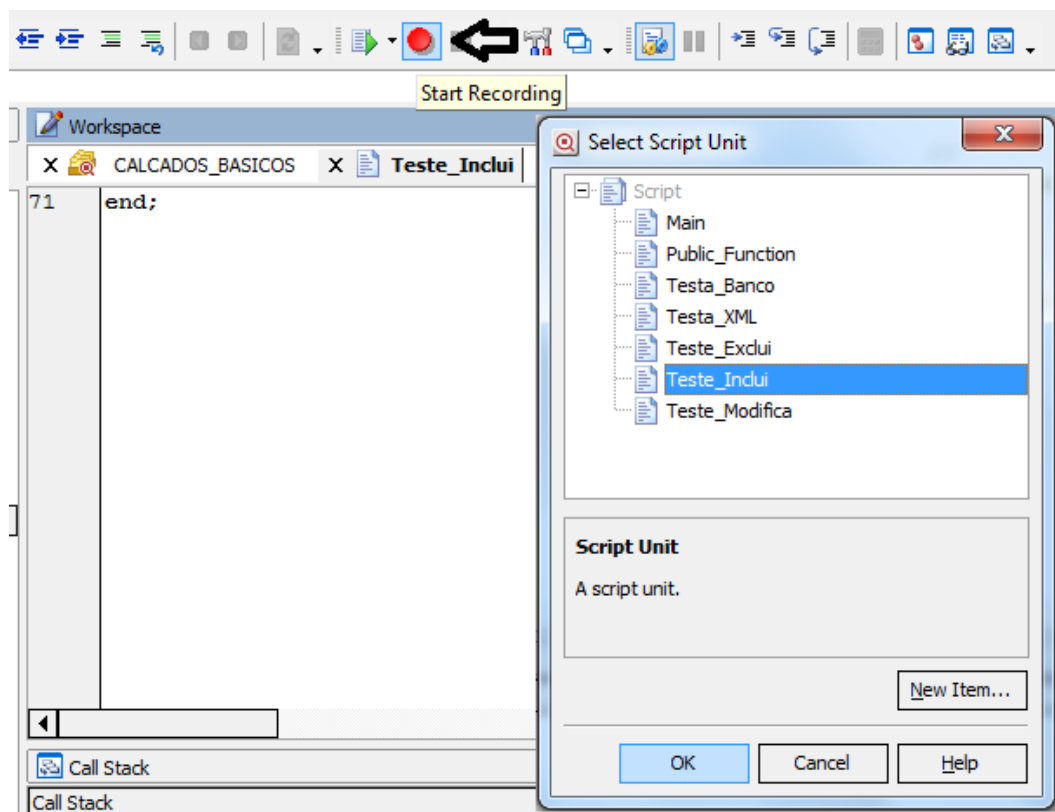


Figura 22 - TestComplete start recording.
Fonte: Elaborada pelo autor.

A função *Start Recording* cria uma macro de gravação das ações do usuário enquanto o usuário estiver usando a aplicação, depois transforma estas ações para o *script* em comandos de linguagem, neste caso na linguagem *Delphi*. Na gravação das ações é possível capturar os controles do objeto, capturar a tela, pausar a gravação e dar um *stop* na gravação, o *stop* para definitivamente a gravação das ações e transforma em comandos de linguagem, conforme apresenta-se na figura 23 e 24 respectivamente:

The screenshot shows a software application window titled 'TAB.2.Empresas'. At the top, there is a 'Recording' toolbar with various icons and a 'Stop' button. The main area of the window is divided into two tabs: '1.Dados I' and '2.Duplicatas/SPED'. The '1.Dados I' tab is active, displaying a form for company information. The form includes fields for Holding (0001), Nome (TESTCOMPLETE NOME EMPRESA), Endereço (RUA MARQUÊS DO PARANÁ), Número (251), Complemento (teste COMPLE), Bairro (IDEAL), CEP (93336-250), Cidade (0093300), NOVO HAMBURGO - RS, Telefone (51) 3594-1000, Fax (51) 3594-1000, Email (luis.paulo@genesysnh.com.br), Site (www.genesysnh.com.br), CNPJ (73.795.486/0001-75), CNPJ SERASA, Inscrição Estadual (161/0036686), Logotipo (\FONTES-TestComplete\ARQUIVOS\TESTCOMPLETE2.jpg), Código ACI (0123), % IRRF por nota fiscal do FAT (1,00), Identificação p/ o sistema (TESTCOMPLETE), Nome completo (TESTCOMPLETE AUTOMACAO DE TEST), Número NIRE, Aliquota PIS (1,65), Tipo de apuração (1.Lucro Real), and Aliquota COFINS (7,60). To the right of the form is a section titled 'Número da última ordem de compra' with a table of order numbers generated by different methods. Below this table is a 'Data registro' field (18/11/2013) and two checkboxes for 'Cooperativa' and 'Desativado'. At the bottom of the window is a menu bar with icons and labels for 'Incluir', 'Modificar', 'Consultar', 'Salvar', 'Desfazer', 'Excluir', 'Auxílio', 'Imprimir', 'Browse', and 'Fechar'. The status bar at the very bottom shows 'Genesys Engenharia de Software Ltda' and 'Incluindo'.

Número da última ordem de compra	
Gerada por remessa	00000010
Gerada por entrada pedido	00000009
Gerada por requisição	00000008
Gerada manual	00000007
Gerada por compra antecipada	00000006
Gerada por projeção	00000005

Figura 23 - TestComplete Start Recording ações do usuário.
Fonte: Elaborada pelo autor.

```

Workspace
Teste_Inclui *
52 procedure Test1;
53   var w1 : OleVariant;
54   var w2 : OleVariant;
55 begin
56   w1 := Sys.Process('gmax').Window('TfmTab000002', 'TAB.2.EMPRESAS');
57   w1.Window('TPanel').Window('TBitBtn', '&Modificar').Keys('~i');
58   w1.Window('TGenMaskEdit', ' ').Keys('6[Enter]');
59   w2 := w1.Window('TPageControl');
60   w2.Keys('[Tab]');
61   w2.ClickTab('&1.Dados I');
62   w1 := w2.Window('TTabSheet', '&1.Dados I');
63   w1.Window('TGenDbEdit', ' ').Keys('1[Enter]');
64   w1.Window('TGenDbEdit', '', 26).Keys('TESTCOMPLETE NOME EMPRESA[Enter]');
65   w1.Window('TGenDbEdit', '', 25).Keys('RUA MARQU! [D222]!eS DO PARAN[D219]!a[Enter]');
66   w1.Window('TGenDbEdit', '', 7).Keys('251[Enter]');
67   w1.Window('TGenDbEdit', '', 1).Keys('teste COMPLE[Enter]');
68   w1.Window('TGenDbEdit', '', 24).Keys('IDEAL[Enter]');
69   w1.Window('TGenDbEdit', ' - ').Keys('93336250[Enter]');
70   w1.Window('TGenDbEdit', ' ').Keys('0093300[Enter]');
71   w1.Window('TGenDbEdit', '*', 21).Keys('5135941000[Enter]');
72   w1.Window('TGenDbEdit', '*', 19).Keys('[Enter]');
73   w1.Window('TGenDbEdit', '*', 20).Keys('513594100[Enter]');
74   w1.Window('TGenDbEdit', '*', 18).Keys('[Enter]');
75   w1.Window('TGenDbEdit', '', 17).Keys('luis.paulo@genesysnh.com.br[Enter]');
76   w1.Window('TGenDbEdit', '', 8).Keys('www.genesysnh.com.br[Enter]');
77   w1.Window('TGenDbEdit', '', 16).Keys('73795486000175[Enter]');
78   w1.Window('TGenDbEdit', '', 4).Keys('[Enter]');
79   w1.Window('TGenDbEdit', '', 15).Keys('1610036686[Enter]');
80   w1.Window('TGenDbEdit', '', 13).Keys('C:\FONTES-TestComplete\ARQUIVOS\TESTCOMPLETE2.jpg[Enter]');
81   w1.Window('TGenDbEdit', '', 12).Keys('0123[Enter]');
82   w1.Window('TGenDbEdit', '', 5).Keys('1,00[Enter]');
83   w1.Window('TGenDbEdit', '', 11).Keys('TESTCOMPLETE[Enter]');
84   w1.Window('TGenDbEdit', '', 10).Keys('TESTCOMPLETE AUTOMACAO DE TEST[Enter]');
85   w1.Window('TGenDbEdit', '', 6).Keys('[Enter]');
86   w1.Window('TGenDbComboBox', '', 2).Keys('1[Enter]');
87   w1.Window('TGenDbEdit', '', 3).Keys('1,65[Enter]');
88   w1.Window('TGenDbEdit', '', 2).Keys('7,60');
89 end;

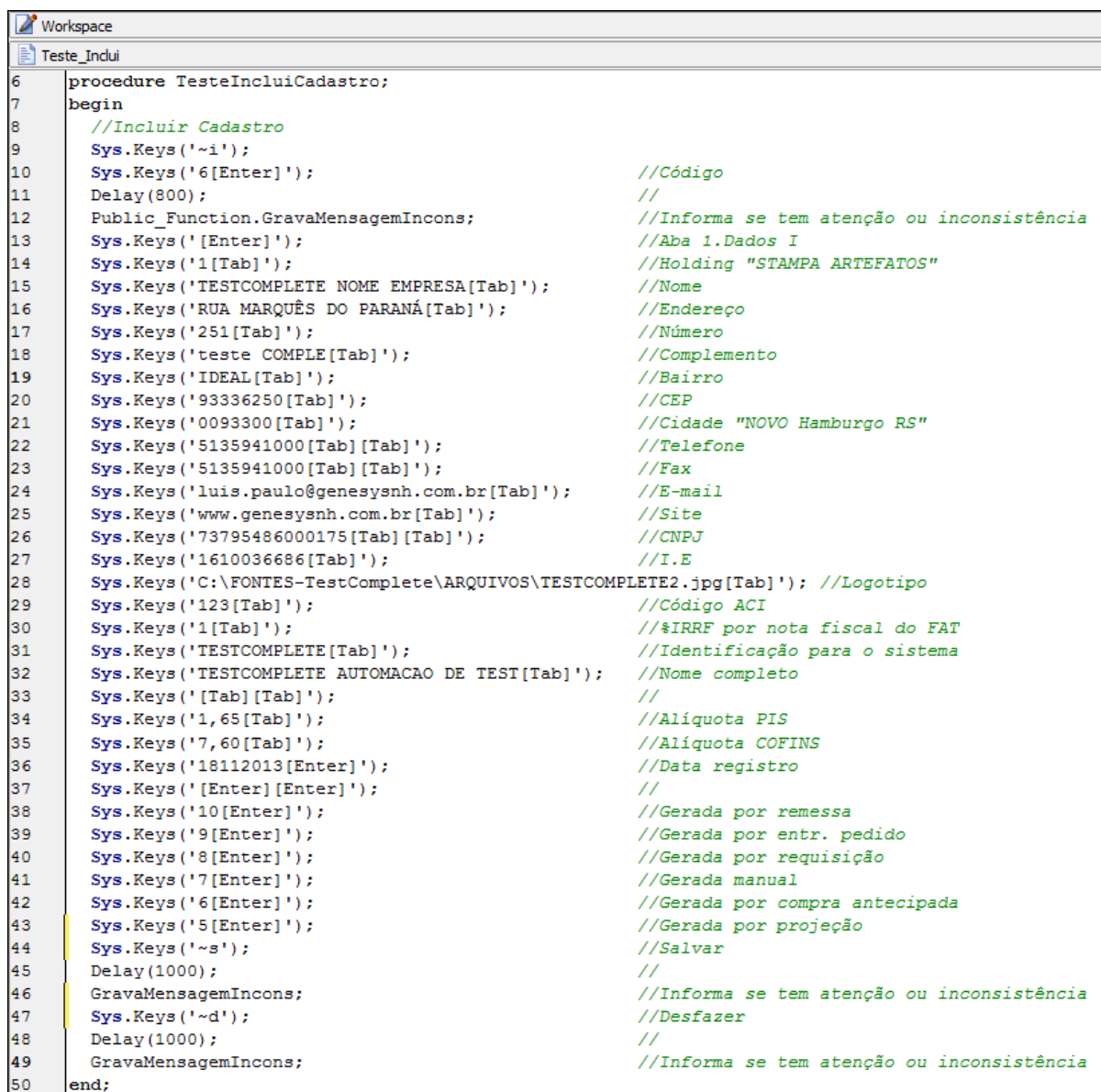
```

Figura 24 - Script de teste IncluiCadastro criado na linguagem DelphiScript.
Fonte: Elaborada pelo autor.

Para todas a teclas digitadas no teclado e objeto evidenciado o *Testcomplete* cria uma variável do tipo *OleVariant*, pode ser observado na figura acima que a variável é o w1 e w2. Note que a ferramenta captura os objetos e componentes destes objetos, objetos sendo este o programa e o formulário, portanto nesta figura acima podemos classificar os seguintes métodos:

- **Sys.Process('gmax')**: Refere-se ao objeto ativo, sendo a aplicação;
- **Window('TfmTab000002', 'TAB.2.EMPRESAS')**: Refere-se ao objeto ativo, sendo o formulário;
- **Os demais que possuem Window('Etc..')**: Refere-se aos componentes do formulário TAB.2.EMPRESAS;
- **Keys('...')**: Refere-se a entrada de valores pelo teclado.

Como este *script* de inclusão deve possuir apenas a entrada de valores do teclado, foi reduzido o código gerado utilizando o método `Sys.Keys()`, este método serve para referenciar-se a entrada de valores por teclas, foi retirado o método gerado `Sys.process('gmax')`, este método já é utilizado na *procedure* **AbreTelaRotina**, e também foi retirado o métodos dos componentes. Essa redução de código reduz o processamento na execução dos testes, facilita a leitura do código e exclui a necessidade de declarar as variáveis, conforme pode-se observar na figura 25:



```

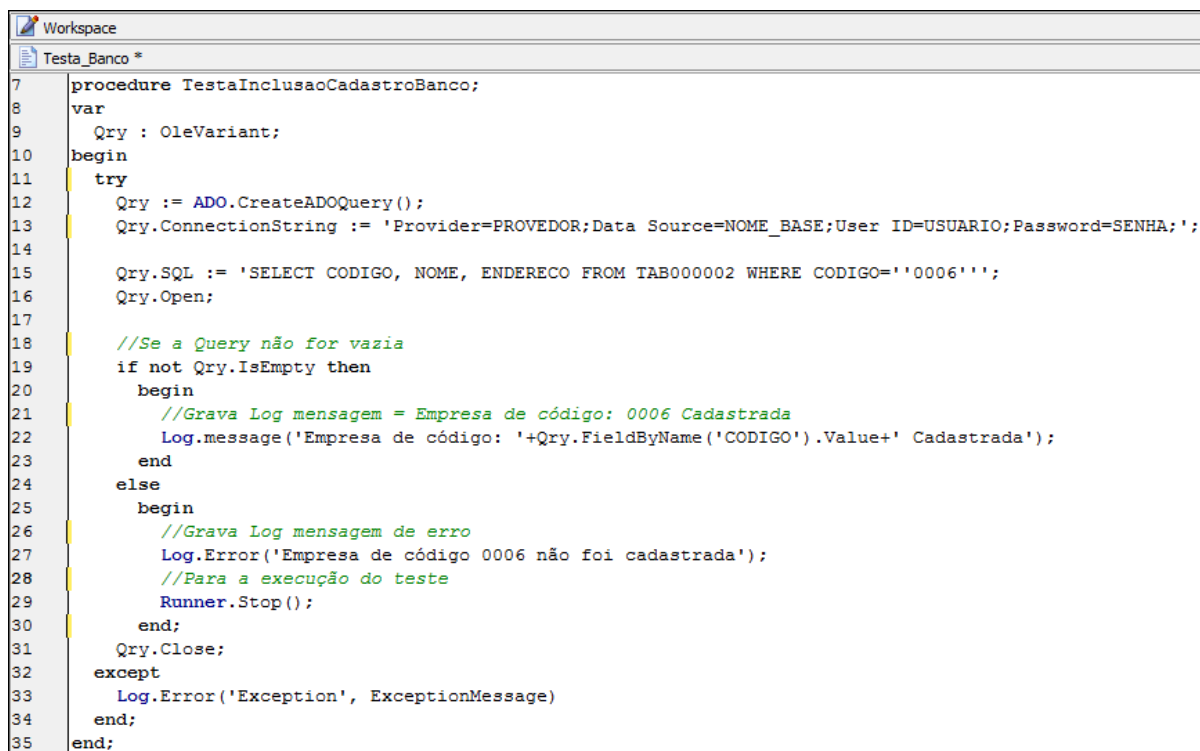
6  procedure TesteIncluiCadastro;
7  begin
8      //Incluir Cadastro
9      Sys.Keys('~i');
10     Sys.Keys('6[Enter]');           //Código
11     Delay(800);                     //
12     Public_Function.GravaMensagemIncons; //Informa se tem atenção ou inconsistência
13     Sys.Keys('[Enter]');           //Aba 1.Dados I
14     Sys.Keys('1[Tab]');             //Holding "STAMPA ARTEFATOS"
15     Sys.Keys('TESTCOMPLETE NOME EMPRESA[Tab]'); //Nome
16     Sys.Keys('RUA MARQUÊS DO PARANÁ[Tab]'); //Endereço
17     Sys.Keys('251[Tab]');           //Número
18     Sys.Keys('teste COMPLETE[Tab]'); //Complemento
19     Sys.Keys('IDEAL[Tab]');          //Bairro
20     Sys.Keys('93336250[Tab]');       //CEP
21     Sys.Keys('0093300[Tab]');        //Cidade "NOVO Hamburgo RS"
22     Sys.Keys('5135941000[Tab][Tab]'); //Telefone
23     Sys.Keys('5135941000[Tab][Tab]'); //Fax
24     Sys.Keys('luis.paulo@genesysnh.com.br[Tab]'); //E-mail
25     Sys.Keys('www.genesysnh.com.br[Tab]'); //Site
26     Sys.Keys('73795486000175[Tab][Tab]'); //CNPJ
27     Sys.Keys('1610036686[Tab]');    //I.E
28     Sys.Keys('C:\FONTES-TestComplete\ARQUIVOS\TESTCOMPLETE2.jpg[Tab]'); //Logotipo
29     Sys.Keys('123[Tab]');           //Código ACI
30     Sys.Keys('1[Tab]');             //Irrf por nota fiscal do FAT
31     Sys.Keys('TESTCOMPLETE[Tab]'); //Identificação para o sistema
32     Sys.Keys('TESTCOMPLETE AUTOMACAO DE TEST[Tab]'); //Nome completo
33     Sys.Keys('[Tab][Tab]');         //
34     Sys.Keys('1,65[Tab]');          //Alíquota PIS
35     Sys.Keys('7,60[Tab]');          //Alíquota COFINS
36     Sys.Keys('18112013[Enter]');    //Data registro
37     Sys.Keys('[Enter][Enter]');     //
38     Sys.Keys('10[Enter]');          //Gerada por remessa
39     Sys.Keys('9[Enter]');           //Gerada por entr. pedido
40     Sys.Keys('8[Enter]');           //Gerada por requisição
41     Sys.Keys('7[Enter]');           //Gerada manual
42     Sys.Keys('6[Enter]');           //Gerada por compra antecipada
43     Sys.Keys('5[Enter]');           //Gerada por projeção
44     Sys.Keys('~s');                 //Salvar
45     Delay(1000);                   //
46     GravaMensagemIncons;           //Informa se tem atenção ou inconsistência
47     Sys.Keys('~d');                 //Desfazer
48     Delay(1000);                   //
49     GravaMensagemIncons;           //Informa se tem atenção ou inconsistência
50 end;
  
```

Figura 25 - Redução do script de teste IncluirCadastro.
Fonte: Elaborada pelo autor.

Após rodar o *script* de cadastro da empresa é preciso verificar se a empresa foi salva no banco de dados, para isto foi criado um novo *script* nomeado de Testa_Banco, este *script* não possui gravação de ações do usuário e sim métodos de conexão com o banco de dados e verificação de dados do banco de dados. Neste *script* foi criado três *procedure*'s de verificação:

- *Procedure TestaInclusaoCadastroBanco;*
- *Procedure TestaModificacaoCadastroBanco;*
- *Procedure TestaExclusaoCadastroBanco;*

A *procedure TestaInclusaoCadastroBanco* é utilizada para verificação do cadastro da empresa, *procedure* esta ilustrada na figura 26:



```

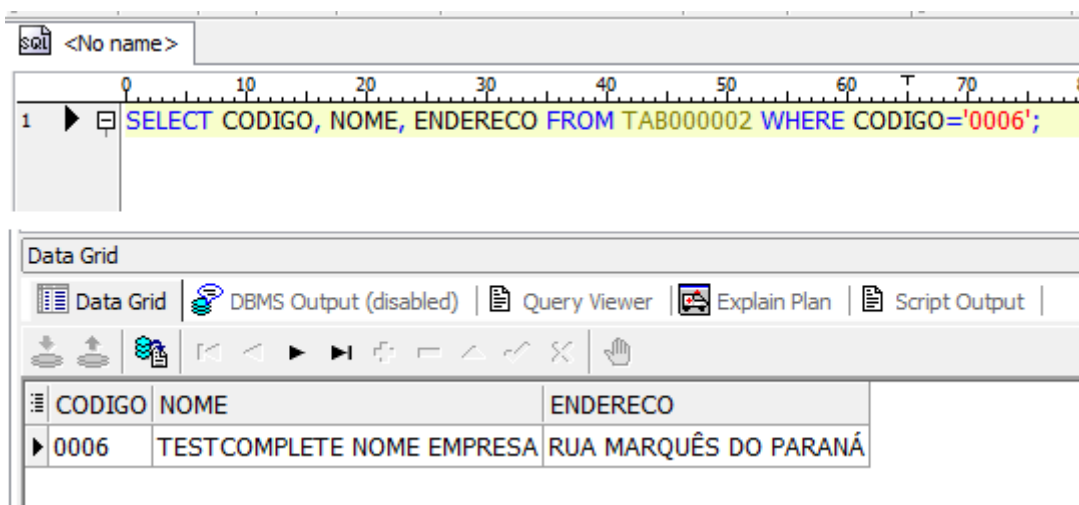
7 procedure TestaInclusaoCadastroBanco;
8 var
9     Qry : OleVariant;
10 begin
11     try
12         Qry := ADO.CreateADOQuery();
13         Qry.ConnectionString := 'Provider=PROVEDOR;Data Source=NOME_BASE;User ID=USUARIO;Password=SENHA;';
14
15         Qry.SQL := 'SELECT CODIGO, NOME, ENDEREÇO FROM TAB000002 WHERE CODIGO='''0006''';
16         Qry.Open;
17
18         //Se a Query não for vazia
19         if not Qry.IsEmpty then
20             begin
21                 //Grava Log mensagem = Empresa de código: 0006 Cadastrada
22                 Log.message('Empresa de código: '+Qry.FieldName('CODIGO').Value+' Cadastrada');
23             end
24         else
25             begin
26                 //Grava Log mensagem de erro
27                 Log.Error('Empresa de código 0006 não foi cadastrada');
28                 //Para a execução do teste
29                 Runner.Stop();
30             end;
31         Qry.Close;
32     except
33         Log.Error('Exception', ExceptionMessage)
34     end;
35 end;

```

Figura 26 - Procedure TestaInclusaoCadastroBanco.
Fonte: Elaborada pelo autor.

Esta *procedure* realiza a conexão com o banco de dados pelo método *ADO.CreateADOQuery* e faz um *select* na tabela TAB000002, onde o valor do campo CODIGO é igual a 0006, código da empresa cadastrada. Seria semelhante ao efetuar um *select* no próprio banco de dados através do software Toad para banco de dados, porém este *select* do *TestComplete* não apresenta fisicamente a

tabela e seus valores. Na figura 27 segue este mesmo *select* rodado no software Toad for Oracle apresentando fisicamente a tabela e seus valores:



The screenshot shows the Toad for Oracle interface. At the top, a SQL editor contains the query: `SELECT CODIGO, NOME, ENDERECO FROM TAB000002 WHERE CODIGO='0006';`. Below the editor is a 'Data Grid' window. It has tabs for 'Data Grid', 'DBMS Output (disabled)', 'Query Viewer', 'Explain Plan', and 'Script Output'. The 'Data Grid' tab is active, displaying the results of the query in a table with three columns: CODIGO, NOME, and ENDERECO. The first row shows the values '0006', 'TESTCOMPLETE NOME EMPRESA', and 'RUA MARQUÊS DO PARANÁ'.

CODIGO	NOME	ENDERECO
0006	TESTCOMPLETE NOME EMPRESA	RUA MARQUÊS DO PARANÁ

Figura 27 - Select da tabela TAB000002 - TestaInclusaoCadastroBanco.
Fonte: Elaborada pelo autor.

Seguidamente esta *procedure* verifica se a *query* não está vazia, se contém dados, significando que a empresa foi cadastrada no banco de dados, se possuir dados gera um *Log.message* de que a empresa 0006 foi cadastrada, se não tiver dados gera um *Log.Error* de que a empresa não foi cadastrada e para a execução do teste pelo método *Runner.Stop*.

Próxima rotina de teste é a alteração da empresa cadastrada, nesta alteração foi alterado o campo nome, de TESTCOMPLETE NOME EMPRESA para NOME DA EMPRESA TC, após criar o *script* de alteração e rodá-lo, foi efetuado o teste da *procedure* **TestaModificacaoCadastroBanco**, que contém a mesma conexão de banco de dados e mesmo *select* da *procedure* anterior, alterando apenas a condição conforme ilustrado na figura 28:

```

47
48 //Se o valor do campo (NOME) do Banco de Dados for igual a NOME DA EMPRESA TC
49 if (Qry.FieldByName('NOME').Value = 'NOME DA EMPRESA TC') then
50     begin
51         //Grava Log mensagem de que o nome da empresa foi alterado para NOME DA EMPRESA TC
52         Log.message('Alterado o nome da empresa 0006 para: '+Qry.FieldByName('NOME').Value);
53     end
54 else
55     begin
56         //Grava Log mensagem de erro
57         Log.Error('Nome da empresa não foi alterado para: NOME DA EMPRESA TC');
58         //Para a execução do teste
59         Runner.Stop();
60     end;
61 Qry.Close;
62 except
63     Log.Error('Exception', ExceptionMessage)
64 end;
65 end;

```


Figura 28 - Condição - Procedure TestaModificacaoCadastroBanco.
Fonte: Elaborada pelo autor.

Por fim a rotina de teste de exclusão da empresa, após criar o *script* de exclusão e rodá-lo, foi efetuado o teste da *procedure* **TestaExclusaoCadastroBanco** que contém a mesma conexão de banco de dados e mesmo *select* das *procedure's* anteriores, alterando apenas a condição conforme ilustrado na figura 29:

```

78      //Se a Query for vazia
79      if Qry.IsEmpty then
80          begin
81              //Grava Log mensagem
82              Log.Message('Empresa de código: 0006 Foi excluída com sucesso');
83          end
84      else
85          begin
86              //Grava Log mensagem de erro
87              Log.Error('Empresa ' + IntToStr(Qry.FieldName('CODIGO').Value) + ' Não foi excluída');
88              //Para a execução do teste
89              Runner.Stop();
90          end;
91      Qry.Close;
92      except
93          Log.Error('Exception', ExceptionMessage)
94      end;
95  end;

```

Figura 29 - Condição - Procedure TestaExclusaoCadastroBanco.
Fonte: Elaborada pelo autor.

Esta condição acima é semelhante a da *procedure* **TestaInclusaoCadastroBanco**, o que diferencia uma da outra é que no *if* não possui o **not**, significando que se a *query* for vazia vai estar correto, ao contrário da outra que verificava se a *query* tinha dados, pois trata-se de uma exclusão onde é deletado os dados da tabela, conforme pode-se observar na figura 30 que a tabela está vazia ao rodar o mesmo *select*.

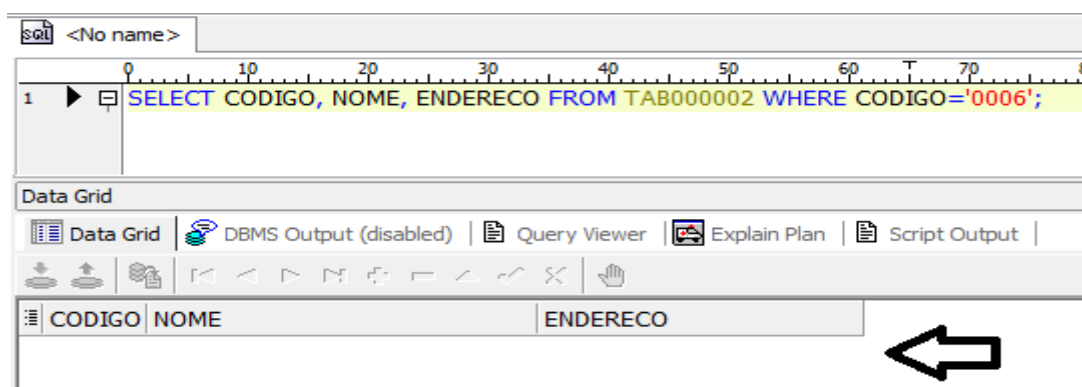


Figura 30 - Select da tabela TAB000002 - TestaExclusaoCadastroBanco.
Fonte: Elaborada pelo autor.

Criado um novo projeto de nome Relatorio_ATB_3_2 para a rotina de teste do formulário ATB.3.2.Remessas bancárias, onde é efetuado a impressão do relatório Planilha Vendedor - Banco - Modelo 1. formulário este apresentado na figura 31 logo abaixo.

Figura 31 - Formulário ATB.3.2.Remessas bancárias.
Fonte: Elaborada pelo autor.

Neste projeto também foi adicionado o *script Public_Function* e por padrão o *script Main*. Criado *script* com nome Testa_Relatorio para o teste do formulário. Note que na figura 32 logo abaixo é declarado as seguintes variáveis onde cada uma recebe um método:

- Variável **Sistema**: Recebe a aplicação, `Sys.process('gmax');`
- Variável **Formulario**: Recebe a variável Sistema mais o formulário, `Window('TfmAtb30002', 'ATB.3.2.Remessas bancárias', 1);`
- Variável **ComboBox_Modelo**: Recebe a variável Formulario mais o *comboBox*, `Window('TGenDbComboBox', '', 1);`
- Variável **Loop1**: É declarado um *arrey* do tipo **for** onde é digitado 8 vezes a tecla 1 mais o *Tab*.

```

5 procedure TestaRelatorio_ATB_3_2;
6 var
7     Sistema; Formulario; ComboBox_Modelo;
8     CheckBox_Imp_Dados_Cadastrais;
9     Loop1;
10 begin
11     //Obtém o processo de aplicação, o formulário eo controle ComboBox
12     Sistema := Sys.Process('gmax');
13     Formulario := Sistema.Window('TfmAtb030002', 'ATB.3.2.Remessas bancárias', 1);
14     ComboBox_Modelo := Formulario.Window('TGenDbComboBox', '', 1);
15     //Seleciona o ComboBox de index 0
16     ComboBox_Modelo.ClickItem(0);           //Modelo "1"
17     //Grava log do ComboBox selecionado
18     Sys.Clipboard := ComboBox_Modelo.wText;
19     Log.Message(Sys.Clipboard);
20     Sys.Keys('[Tab]');                       //
21     Sys.Keys('28112013[Tab]');               //Data inicial
22     Sys.Keys('31122013[Tab]');               //Data final
23     Sys.Keys('28112013[Tab]');               //Vencimento inicial
24     Sys.Keys('31122013[Tab]');               //Vencimento final
25     Sys.Keys('000001[Tab]');                 //Remessa inicial
26     Sys.Keys('999999[Tab]');                 //Remessa final
27     CheckBox_Imp_Dados_Cadastrais := Sys.Process('gmax').Window('TfmAtb030002', 'ATB.3.
28     //Se o CheckBox Imprime dados cadastrais estiver desmarcado, vai marcá-lo.
29     if CheckBox_Imp_Dados_Cadastrais.wState = cbUnchecked then
30         begin
31             CheckBox_Imp_Dados_Cadastrais.Click;
32         end;
33     Sys.Keys('[Tab]');                       //
34     //Arrei p/ repetir 8 vezes o dígito 1 + Tab.
35     for Loop1 := 1 to 8 do
36         begin
37             Sys.Keys('1[Tab]');               //Seleção e Ordens
38         end;
39     Sys.Keys('~i');                           //Imprimir
40     Public_Function.AguardaOpc('IMP');        //Aguarda"Dispositivo de saída do relatório"
41     Sys.Keys('~v');                           //Video
42     Sys.Keys('~o');                           //OK
43     BuiltIn.Delay(5000, 'Carregando relatório...'); //Atrasa execução do teste
44     //Tira print da tela e salva no Stores
45     SalvaImagemRegions('ATB_3_2Remessas Bancarias');
46     //Compara a imagem salva no Stores com a tela ativa.
47     CompareRegionsSalvas('ATB_3_2Remessas Bancarias');
48     Sys.Keys('~f');                           //Fechar relatório
49     Sys.Keys('~r');                           //Fechar o formulário
50 end;

```

Figura 32 - Script - procedure TestaRelatorio_ATB_3_2.

Fonte: Elaborada pelo autor.

É possível notar que na figura acima a variável **ComboBox_Modelo** também recebe o método *ClickItem(0)*, que irá selecionar o **Modelo 1**. Planilha Vendedor para o banco, o zero dentro dos parênteses significa o *index* da opção do *ComboBox* Modelo. Como o *CheckBox Imprime dados cadastrais* deve estar sempre marcado quando imprimir o relatório, foi utilizada uma funcionalidade e método que o *TestComplete* fornece, funcionalidade esta que se encontra em *Object Properties*,

ao clicar na função irá abrir uma janela onde é selecionado o *Finder Tool* e arrastado para o campo desejado, a ferramenta irá mapear todas as propriedades do componente **CheckBox Imprime dados cadastrais**, conforme ilustrado na figura 33:

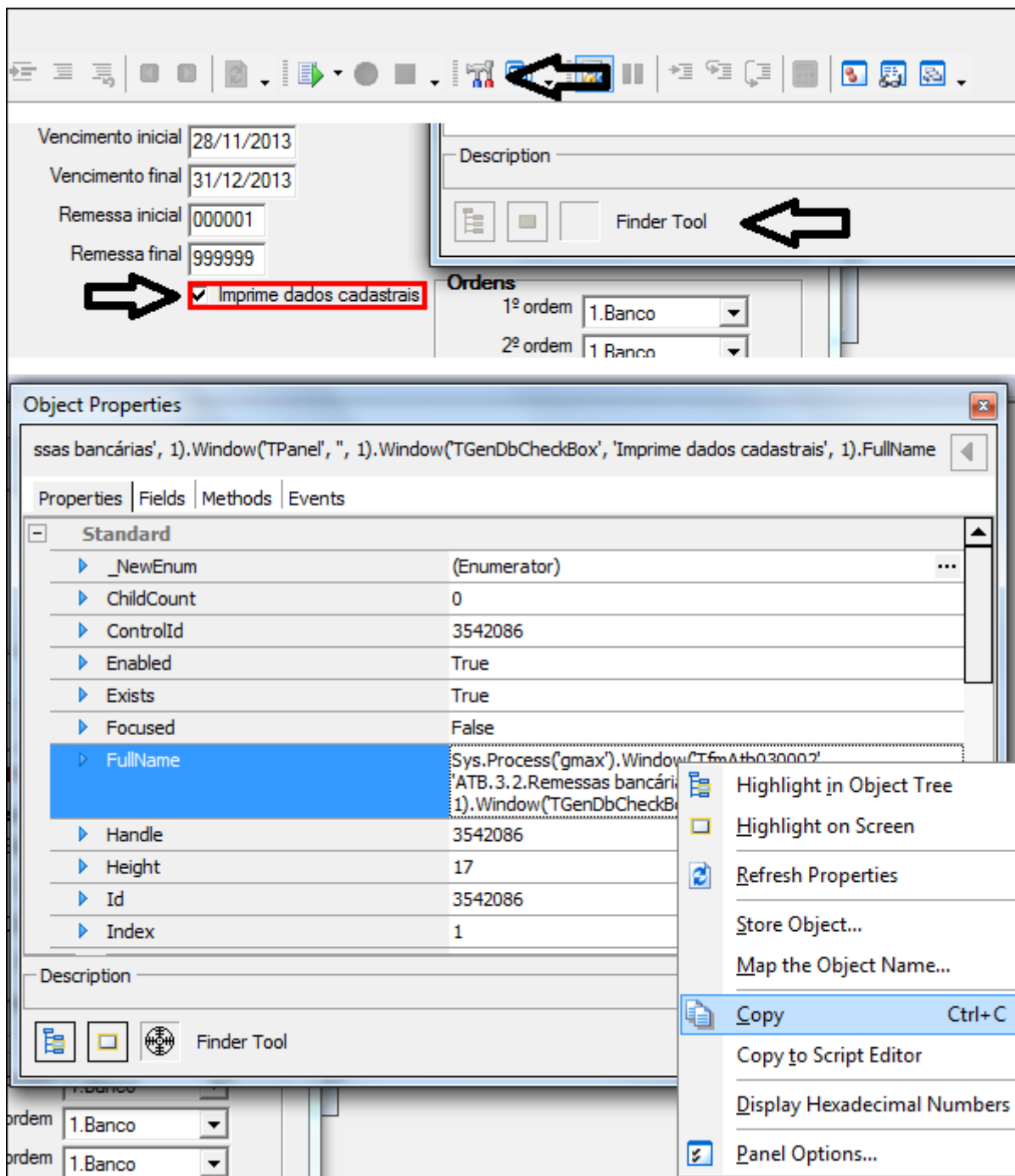


Figura 33 - Object Properties - CheckBox Imprime dados cadastrais.
Fonte: Elaborada pelo autor.

Ao efetuar o mapeamento do *CheckBox* é possível copiar o nome completo deste componente, conforme é demonstrado na figura acima, a variável **CheckBox_Imp_Dados_Cadastrais** declarada junto com as demais variáveis irá

receber o nome completo do *CheckBox* mapeado. Assim que a variável recebe o nome do *CheckBox* é possível utilizar o método *wState* que irá verificar se o *CheckBox* está desmarcado, se estiver desmarcado irá marcá-lo conforme condição criada no *script*, se já estiver marcado o método desconsidera a condição do *script*, método e condição que pode ser observado na figura 34:

```

27   CheckBox_Imp_Dados_Cadastrais := Sys.Process('gmax').Window('TfmAtb030002', 'ATB.3.
28   //Se o CheckBox Imprime dados cadastrais estiver desmarcado, vai marcá-lo.
29   if CheckBox_Imp_Dados_Cadastrais.wState = cbUnchecked then
30       begin
31           CheckBox_Imp_Dados_Cadastrais.Click;
32       end;

```

Figura 34 - Método *wState* - *CheckBox* Imprime dados cadastrais.
Fonte: Elaborada pelo autor.

Logo depois deste método é declarado o *array For* que já foi mencionado, ao imprimir o relatório é chamado a *procedure* **AguardaOpc('IMP')**, onde a mesma tem a finalidade de aguardar a janela 'Dispositivo de saída do relatório', que enquanto não for apresentada irá gerar o método *delay* de 1500. *Procedure* e janela que apresentam-se na figura 35:

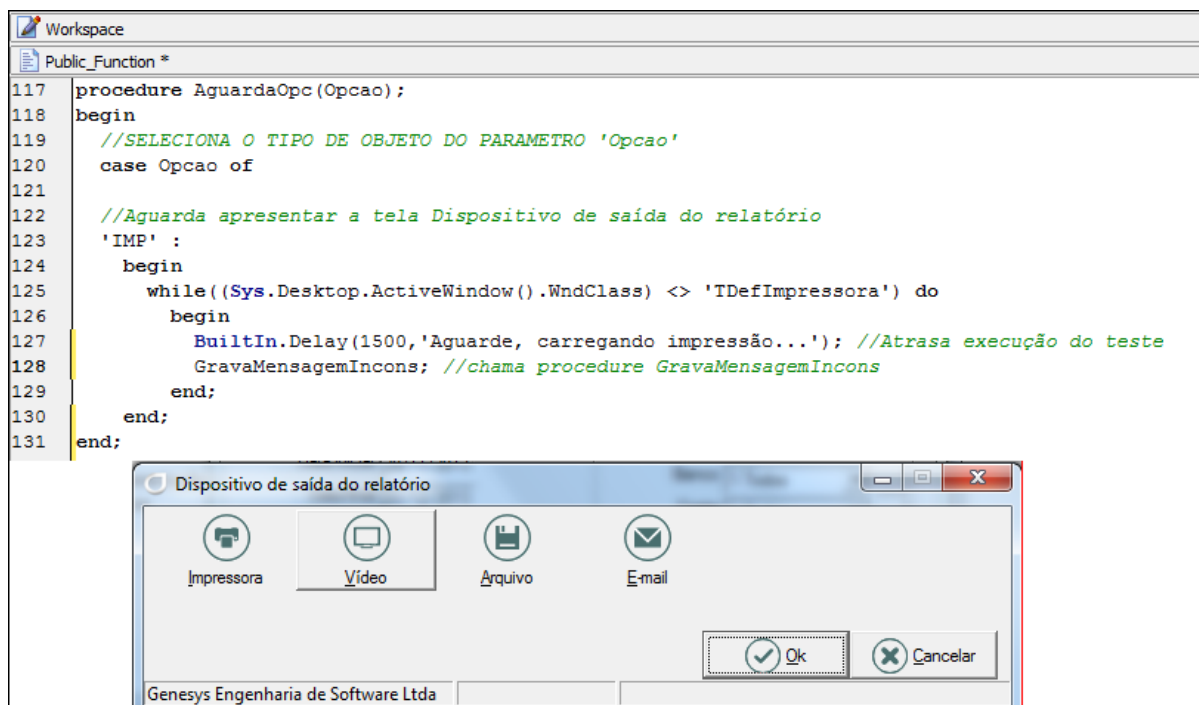


Figura 35 - Dispositivo de saída do relatório
Fonte: Elaborada pelo autor.

Depois de apresentar a janela e sucessivamente o relatório impresso, é chamado as *procedures* **SalvaImagemRegions** e **ComparaRegionsSalvas**, a

procedure **SalvaImagemRegions** tem a finalidade de salvar no item *Stores* do *TestComplete*, item marcado na criação do Project, a imagem da tela, neste caso do relatório. Imagem esta que é salva com o nome de ATB_3_2Remessas_Bancarias.

Na *procedure* **SalvaImagemRegions** é utilizado os métodos *Sys.Desktop.ActiveWindow* e *Regions.AddPicture* que salva a imagem do relatório, *procedure* demonstrada na figura 36 e junto é demonstrado o local onde é salvo a imagem do relatório:

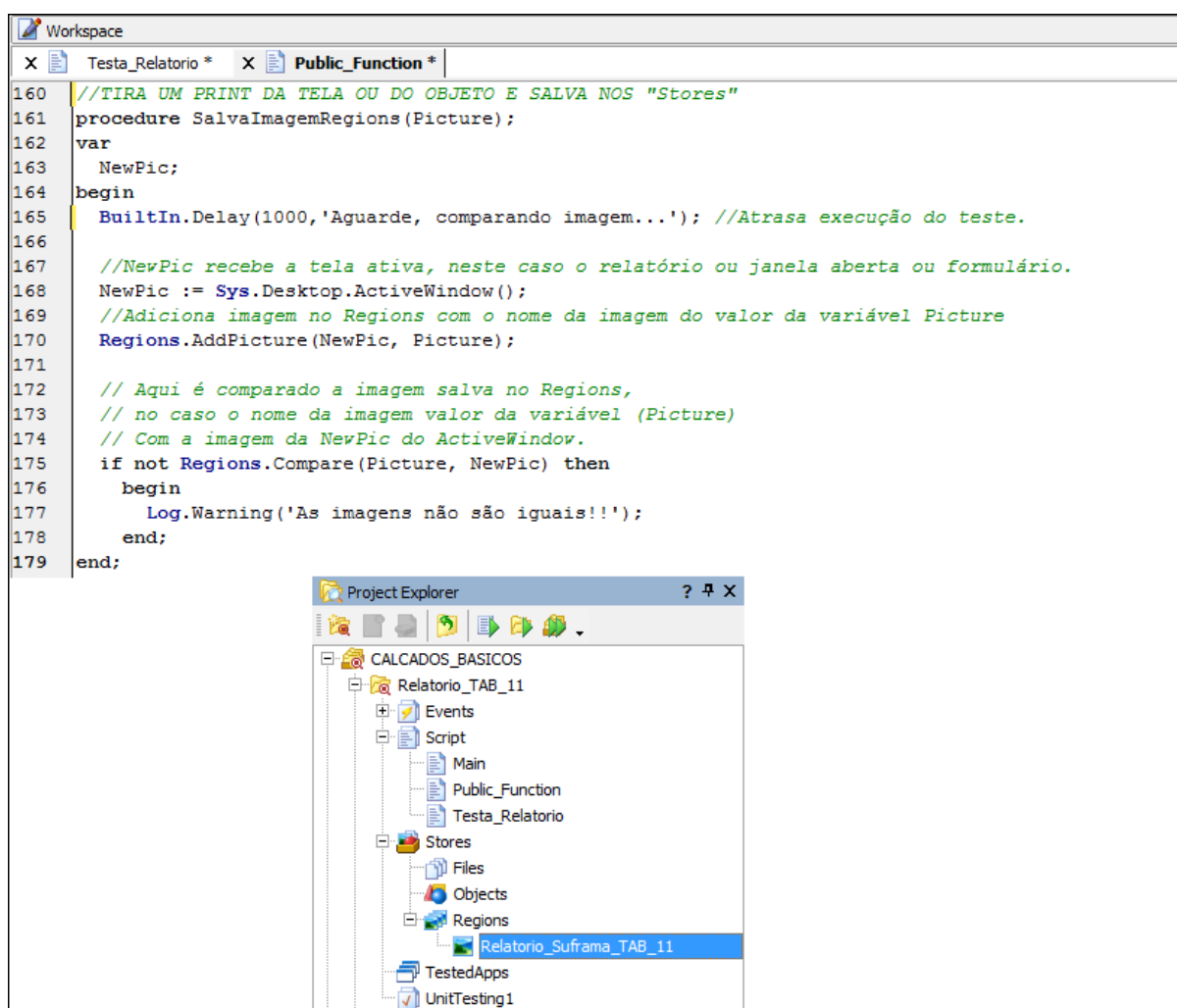


Figura 36 - Salvar Imagem no Stores.
Fonte: Elaborada pelo autor.

A *procedure* **ComparaRegionsSalvas** tem a finalidade de comparar a imagem salva no *Stores* com a imagem da tela ativa, sendo o relatório impresso, comparação que é feita pelo método *Regions.Compare*, se tiver alguma diferença a mesma grava mensagem de *log* pelo método *Log.Warning*('As imagens não são iguais!!'). Conforme pode ser notado na figura 37:

```

181 //Compara a imagem da tela com a imagem salva no "Stores"
182 procedure CompareRegionsSalvas(Picture);
183 var
184     Tela;
185 begin
186     BuiltIn.Delay(1000,'Aguarde, comparando imagem...');
187
188     //Tela recebe a tela ativa, neste caso o relatório ou janela aberta ou formulário.
189     Tela := Sys.Desktop.ActiveWindow();
190
191     // Aqui é comparado a imagem armazenada no Regions,
192     // no caso o nome da imagem valor da variável (Picture)
193     // Com a imagem da Tela do ActiveWindow.
194     if not Regions.Compare(Picture, Tela) then
195     begin
196         Log.Warning('As imagens não são iguais!!');
197     end;
198 end;

```

Figura 37 - Compara Imagens.
Fonte: Elaborada pelo autor.

Se as imagens forem iguais, o *TestComplete* fecha o relatório impresso e o formulário identificado pela rotina “ATB.3.2.Remessas bancárias”, finalizando a execução do teste.

Após ter todo o projeto de automação de testes desenvolvido, o mesmo é executado pela ferramenta *TestComplete*, e com base na visualização de *logs* que a mesma fornece, é possível analisar o resultado das rotinas testadas.

5 RESULTADOS

Ao rodar o *Project Suite* por completo, conforme apresenta-se na figura 38, a ferramenta *Testcomplete* fornece o resultado da execução completa dos *scripts* de teste com os *logs* de execução do teste, resultado este que pode ser observado na figura 39, resultado de execução do teste procedido de forma correta e satisfatória:

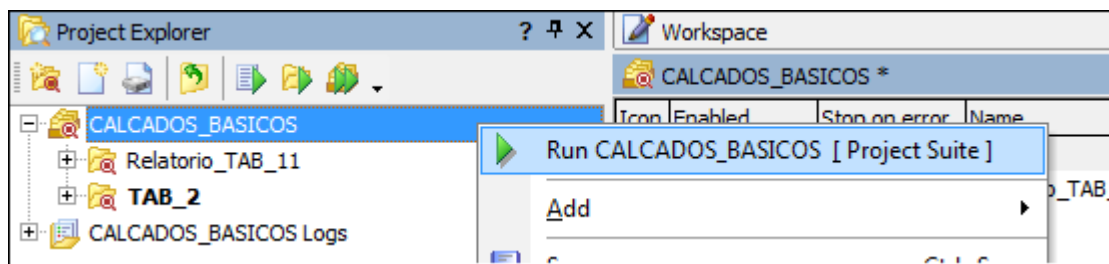


Figura 38 - Execução do Project Suite
Fonte: Elaborada pelo autor.

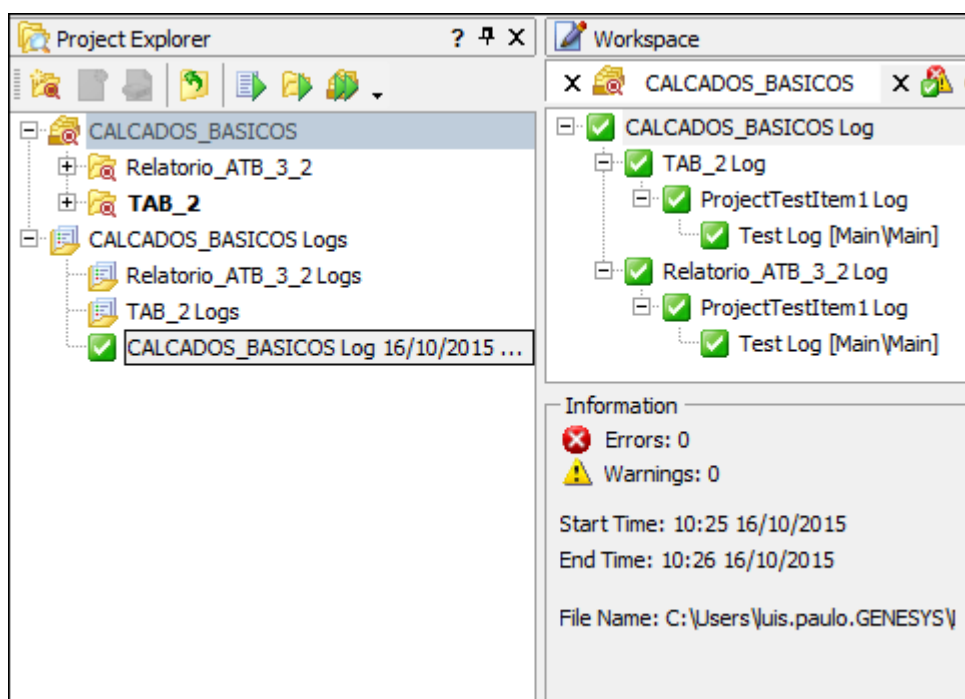


Figura 39 - Resultado da execução do Project Suite
Fonte: Elaborada pelo autor.

Note que na figura 39 no *painel* com nome de *Information* o *Testcomplete* informa os *logs* de *Errors* e *logs* de *Warnings*, como o teste rodou de maneira correta não gerou *logs* de *Errors* e *logs* de *Warnings*, o resultado é zero conforme demonstrado.

Ao forçar um resultado de erro no teste do formulário TAB.2.EMPRESAS, alterando de maneira incorreta o nome da empresa, observe na figura 40 que o *TestComplete* informa o resultado com o *log* de erro que foi tratado na *procedure TestaModificacaoCadastroBanco*, *Log.Error*('Nome da empresa não foi alterado para: NOME DA EMPRESA TC'). Além de gerar o *log* de erro a ferramenta para a execução do teste na linha de código do erro, pois foi inserido logo após o método *Runner.Stop*.

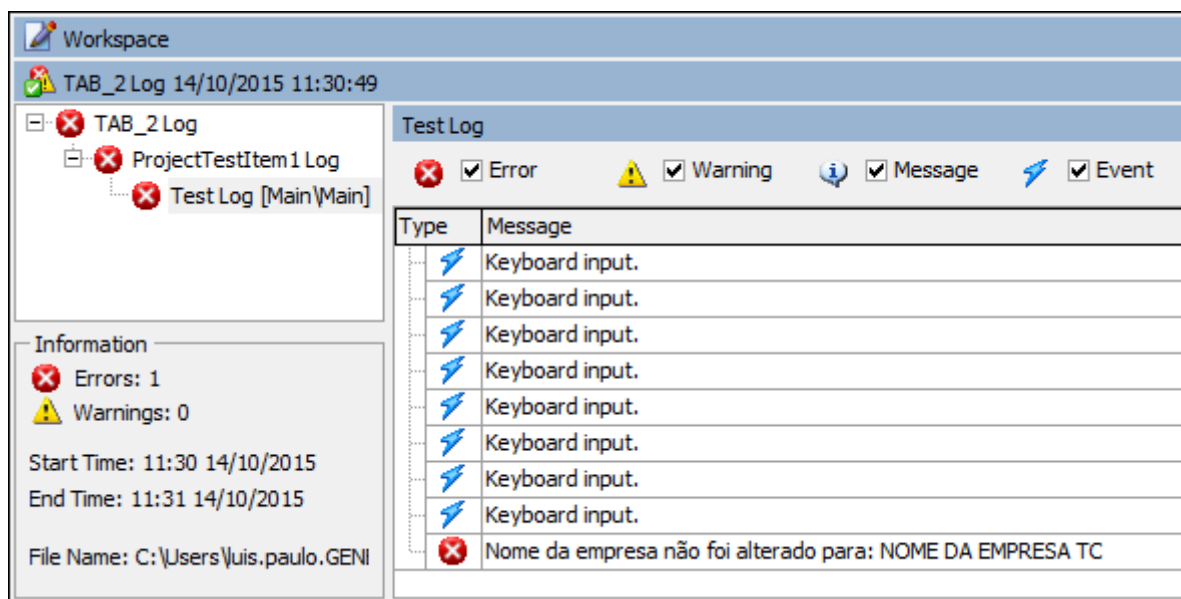


Figura 40 - Erro do teste Project TAB_2
Fonte: Elaborada pelo autor.

Logo após, foi forçado o resultado de erro no teste do formulário ATB.3.2.Remessas bancárias, onde foi desmarcado propositalmente o *checkBox Imprime dados cadastrais*, fazendo com que o resultado da impressão do relatório deste formulário seja diferente da imagem do relatório que foi salva no *Stores*, pois ao chamar neste teste a *procedure CompareRegionsSalvas*, o método *Regions.Compare* gera um *log* de erro de que as imagens não são iguais, conforme pode ser observado na figura 41:

	Regions are not equal.
	The "Picture1" parameter passed to Regions.Compare.
	The "Picture2" parameter passed to Regions.Compare.
	Regions.Compare result.
	As imagens não são iguais!!
	Keyboard input.
	Keyboard input.

Figura 41 - Log de erro das imagens diferentes
Fonte: Elaborada pelo autor.

Além de gerar *log* de erro, o método *Regions.Compare* faz uma comparação pixel á pixel da imagem salva no *Stores* com a imagem temporária que é capturada da tela ativa (relatório impresso). Então o *TestComplete* fornece em uma imagem o resultado diferente da comparação das imagens, conforme é demonstrado na figura 42, resultado do que foi alterado aparece na cor vermelha:

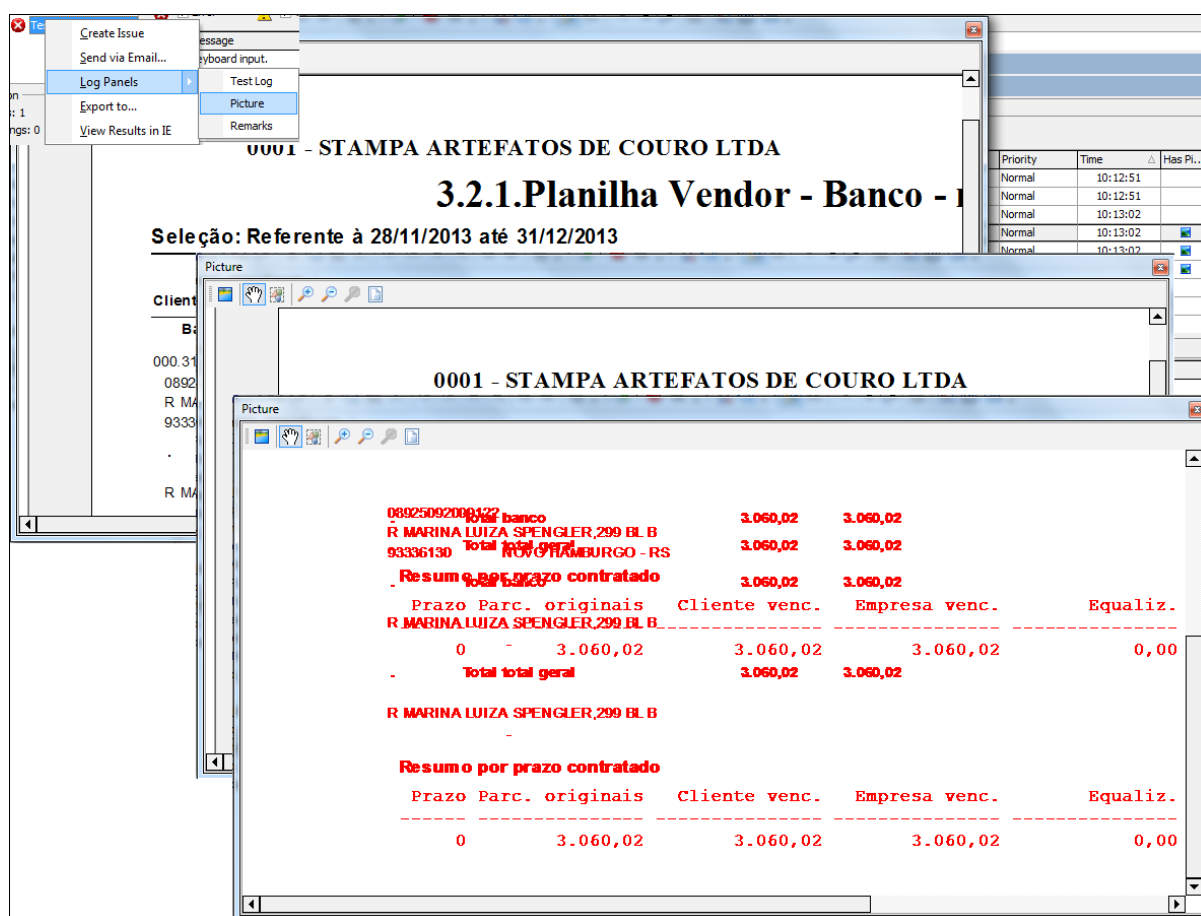


Figura 42 - Resultado das imagens diferentes
Fonte: Elaborada pelo autor.

6 CONCLUSÃO

Após o desenvolvimento e aplicação do projeto na empresa, é possível afirmar que a automação de teste fornecida pela ferramenta *TestComplete*, possui diversos benefícios e vantagens, se comparado com testes manuais.

A tabela 2 apresenta os benefícios em comparação à realização de testes manuais:

Informações	Testes Manuais	Automação de Testes
Tempo de execução do teste	10 minutos à 20 minutos	1 minuto
Precisão do teste	Não aplicável	100%
Qualidade do resultado	Baixa	Alta
Manutenção dos testes	Pouca	Média
Envolvimento humano na execução do teste	100%	0%

Tabela 2 – Benefícios obtidos com a ferramenta *TestComplete*
Fonte: Elaborada pelo autor.

Também é possível afirmar que, através da utilização da ferramenta *TestComplete* para a automação de teste do software ERP GMAX, nos testes foram identificados erros que não seriam capturados se o teste fosse realizado manualmente, pois a ferramenta executa o teste de forma rápida onde a aplicação deve corresponder a agilidade de utilização da mesma, agilidade de teste que não é possível alcançar com testes manuais.

Isto ocasionou a redução de falhas no software, devido a utilização em grande escala.

Falhas de interface gráfica foram encontradas, as quais também não seriam evidenciadas em testes manuais se o testador estivesse distraído, por exemplo, falha na ordem dos componentes (*Tab order*) e falha na máscara dos componentes, como em campos de data-hora, que deviriam conter a devida formatação.

É possível afirmar também que a qualidade visual do software foi melhorada, pois falhas de interface gráfica que eram evidenciadas somente pelo usuário após a homologação, foram descobertas na automação do teste.

O processo de teste de regressão também teve maior qualidade, pois foi possível testar todas as rotinas do sistema, garantindo que novas alterações não

tivessem gerado falhas nas opções já existentes, gerando assim maior confiabilidade no resultado do teste de regressão.

A automação de testes é um investimento com retorno garantido. No entanto, a automação de testes não é solução de todos os problemas de qualidade. A automação de testes não deve ser empregada como um substituto do teste manual, deve ser introduzida como uma técnica adicional, cujo o objetivo principal é agregar valor, sem no entanto, desprezar o teste manual realizado pelos desenvolvedores.

Por fim, para estudos futuros, existe a possibilidade de iniciar novos projetos de testes para todas as plataformas de sistema que a empresa utiliza, como por exemplo os aplicativos *mobile* e suas aplicações *Web*.

7 REFERÊNCIAS

CAETANO, Cristiano. **Automatize os testes de seus sistemas usando o software TestComplete**. [2014]. Disponível em:<<http://www.devmedia.com.br/artigo-clubedelphi-102-testcomplete/11758>>.n Acesso em: 23set. 2015.

CHATTERJEE, Ipsita. **Testing Testability**. [2004]. Disponível em:<<http://www.stickyminds.com/article/testing-testability>>.n Acesso em: 19ago. 2015.

GASPAR, Heloisa. **O que é sistema ERP?** [2012]. Disponível em:<<http://www.pwi.com.br/blog/o-que-e-sistema-erp/>>.n Acesso em 24set. 2015.

ISO/IEC 9126. **Information Technology – software product evaluation – quality characteristics and guidelines for their use**. [2003]. Disponível em:<http://luizcamargo.com.br/arquivos/NBR%20ISO_IEC%209126-1.pdf>.n Acesso em: 29ago. 2015.

LOURENÇÃO, Guilherme e OLIVEIRA, Elisamara. **Testabilidade e planejamento de testes de software**. [2015]. Disponível em:<<http://www.getinews.com.br/volumes/198/testabilidade-e-planejamento-de-testes-de-software>>.n Acesso em: 26ago. 2015.

MYERS, Glenford J. The Art of Software Testing. New Jersey: **John Wiley & Sons Inc.**, 2004.

NETO, Arilo Claudio Dias. **Artigo Engenharia de Software - Introdução a Teste de Software**. [2014]. Disponível em:<<http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035>>.n Acesso em: 23set. 2015.

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

SOMMERVILLE, Ian. **Engenharia de Software**. 8 ed. São Paulo: Pearson Addison-Wesley, 2007.

TANNOURI, Patricia Aline. **O que é Testabilidade?** [2015]. Disponível em:<<http://www.linhadecodigo.com.br/artigo/923/o-que-e-testabilidade.aspx>>.n Acesso em: 26ago. 2015.

VARELA, Marcelo. **Os Procedimentos (Procedures) e Funções (Functions)**. [2002]. Disponível em:<<http://imasters.com.br/artigo/345/linguagens/os-procedimentos-procedures-e-funcoes-functions>>.n Acesso em: 04out. 2015.