|  |
| --- |
| F:\Meus Documentos Jullian\UDESC\5 - Projeto de Software\Trabalho Final\Material Gráfico\logobranco.jpg |

Plano de Testes

Desenvolvimento Stammtisch Control

|  |  |
| --- | --- |
| Professor:  Osmar de Oliveira Braz Júnior | Alunos:  Ederson Jasper  Jullian Hermann Creutzberg |

  Março / 2016

Histórico de Revisões

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 22/02/2016 | 1.0 | Release Inicial | Jullian Hermann Creutzberg |
| 09/03/2016 | 1.1 | Revisão das ferramentas a serem utilizadas | Ederson Jasper |
| 10/03/2016 | 1.2 | Atualização dos Cenários de Testes | Jullian Hermann Creutzberg |
| 12/03/2016 | 1.3 | Atualização dos Cenários de Testes | Jullian Hermann Creutzberg |
| 13/03/2016 | 1.4 | Revisão dos Cenários de Testes | Ederson Jasper |

Índice

1. Introdução 5

1.1 Objetivos 5

1.2 O Stammtisch Control 5

1.3 Escopo 5

1.4 Documentação do Projeto 6

2. Itens de teste 6

2.1 Especificação e Análise de Requisitos 7

2.1.1 Requisitos funcionais 7

2.1.2 Requisitos não-funcionais 7

2.1.3 Casos de Uso 7

2.2 Realização dos testes 8

2.2.1 Teste Unitário 8

2.2.2 Teste de Integração 9

2.2.3 Teste de Sistema e Interface do Usuário 9

2.2.4 Teste do Banco de Dados 9

2.2.5 Teste Funcional 9

2.2.6 Teste do Ciclo de Negócios 9

2.2.7 Teste de Performance 9

2.2.8 Teste de Carga 9

2.2.9 Teste de Stress 10

2.2.10 Teste de Volume 10

2.2.11 Teste de Segurança e de Controle de Acesso 10

2.2.12 Teste de Falha/Recuperação 10

2.2.13 Teste de Instalação 10

3. Premissas 10

4. Análise de riscos 11

5. Ambiente de Testes 12

5.1 Tipos de Teste 13

5.1.1 Teste Unitário 13

5.1.2 Teste de Integração 14

5.1.3 Teste de Sistema e Interface do Usuário 14

5.1.4 Teste de Integridade de Dados e do Banco de Dados 15

5.1.5 Teste de Função 15

5.1.6 Teste de Performance 15

5.1.7 Teste de Carga 16

5.1.8 Teste de Segurança e Controle de Acesso 17

5.1.9 Teste de Instalação 17

5.2 Ferramentas de teste 18

6. Estrutura dos testes 19

6.1 Testes Unitários 19

6.2 Testes de Integração 20

6.3 Testes de Sistema 26

7. Recursos Humanos e Cronograma 30

7.1 Funções e Responsabilidades 30

7.2 Cronograma 31

8. Gerência de defeitos 32

9. Comunicação 32

# Introdução

## Objetivos

Esse documento do Plano de Testes do Software **Stammtisch Control** possui os seguintes objetivos:

* Identificar informações de projeto existentes e os componentes de software que devem ser testados.
* Listar os requisitos a testar recomendados.
* Apresentar os casos de uso desenvolvidos.
* Recomendar e descrever as estratégias de teste a serem empregadas.
* Identificar os recursos necessários e prover uma estimativa dos esforços de teste.
* Listar os elementos resultantes do projeto de testes.
* Identificar os responsáveis pelos testes e delimitar um cronograma de execução.

## O Stammtisch Control

Este sistema provém da necessidade de um software para auxiliar o gerenciamento de um grupo de Stammtisch ao qual os clientes do projeto são coordenadores do grupo.

Entende-se por Stammtisch uma festa popular a qual é realizada anualmente em um determinado local da cidade. Nesta festa, tem-se um agrupamento de pessoas em blocos compostos de 15 a 100 integrantes, em barracas nas quais são consumidos produtos alimentícios e bebidas alcoólicas em geral.

Além disso, cada bloco fica localizado em determinado local cuja identificação é representada pela sua barraca. O valor dos itens necessários para a participação no evento (barraca, camisa, alimentos, bebidas, etc.) é rateado entre os participantes do grupo, sendo que podem existir patrocínios para o bloco. Deve ser considerado que consumo de bebidas alcoólicas é rateado somente entre os que a consomem.

O sistema Stammtisch Control tem como objetivo possibilitar aos coordenadores do grupo gerenciar a participação em eventos, tendo controle de despesas, participantes do grupo, inscritos para os eventos, rateios de despesas dentre outros.

## Escopo

O Stammtisch Control passará pelos testes unitário, de integração e testes de sistema. Os testes unitários e de integração vão lidar com a qualidade funcional, das bases de dados e interface gráfica integrada; enquanto que os testes de sistema tratarão as questões de interfaces quanto ao comportamento do sistema como um todo.

Os testes de performance e os testes de configuração também não serão realizados uma vez que os servidores utilizados serão contratados para uso específico do sistema em nuvem e o provedor se responsabilizará pela configuração e manutenção do serviço.

Para a execução dos testes serão utilizados terminais o mais idênticos possível, em termos de hardware, àquelas que serão utilizadas pelo cliente, a fim de garantir a previsibilidade de performance e compatibilidade.

Os testes mais críticos para este software serão os testes de integração e de sistema, que compõe o maior objetivo deste plano de teste.

Esta primeira versão do plano de testes, abrange as seguintes funcionalidades: o cadastro de participantes dos grupos (REF 01) e a manutenção das categorias de itens (REF 03). Todos os requisitos serão apresentados no item 2, a seguir.

Algumas funcionalidades que não serão testadas neste momento são: envio de e-mail aos participantes, cadastro de itens, cadastro de itens para o evento, geração de lista de compras, o rateio de despesas, o cadastro de patrocínios, controle de pagamento dos participantes e a realização da inscrição do participante e acompanhante nos eventos. Sendo estes testes implementados e realizados em futuras iterações do presente plano.

## Documentação do Projeto

A tabela abaixo identifica a documentação, a disponibilidade e o arquivo ou local onde se encontram as informações utilizadas para desenvolver este plano de testes, baseado no projeto de sistema desenvolvido para o Stammtisch Control.

|  |  |  |
| --- | --- | --- |
| **Documento** | **Disponível** | **Arquivo/Local** |
| Especificação de Requisitos | ◼ Sim 🞏 Não | *Ederson-Jullian-Trabalho-ProjetoSoftware-Final.eap* |
| Plano de Projeto | ◼ Sim 🞏 Não | *Ederson-Jullian-Trabalho-ProjetoSoftware-Final.eap* |
| Modelo de Análise | ◼ Sim 🞏 Não | *Ederson-Jullian-Trabalho-ProjetoSoftware-Final.eap* |
| Modelo de Projeto | ◼ Sim 🞏 Não | *Ederson-Jullian-Trabalho-ProjetoSoftware-Final.eap* |
| Documento de Arquitetura | ◼ Sim 🞏 Não | *Ederson-Jullian-Trabalho-ProjetoSoftware-Final.eap* |
| Protótipo | ◼ Sim 🞏 Não | *Ederson-Jullian-Trabalho-ProjetoSoftware-Final.eap* |
| Manual do Usuário | 🞏 Sim ◼ Não |  |
| Lista de Riscos | ◼ Sim 🞏 Não | Neste documento |

# Itens de teste

Os testes devem cobrir a obrigatoriedade dos campos descritos no levantamento de requisitos, sendo que esta verificação deve ser realizada através da interface para apresentar uma mensagem condizente ao usuário.

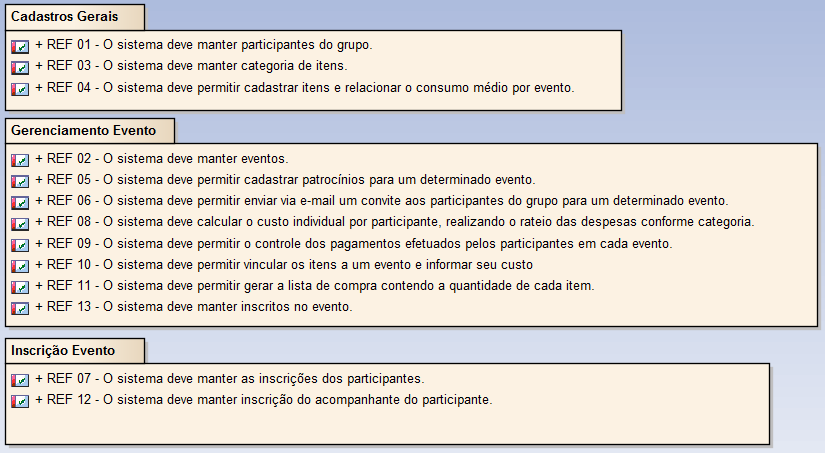
Deve ser testada a inclusão, atualização e exclusão de registros no banco para cada entidade persistente do projeto. Será verificado por meio dos testes se os registros são buscados no banco de dados corretamente conforme os parâmetros informados.

Serão apresentados a seguir os requisitos funcionais e não-funcionais especificados e analisados que compõe o projeto Stammtisch Control. Será demonstrada também a visão geral dos casos de uso resultante da análise dos requisitos.

## Especificação e Análise de Requisitos

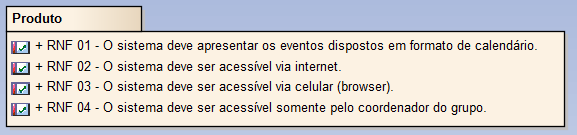
### Requisitos funcionais

A figura abaixo apresenta os requisitos funcionais do projeto de desenvolvimento do sistema Stammtisch Control, sendo agrupados por módulos do sistema.



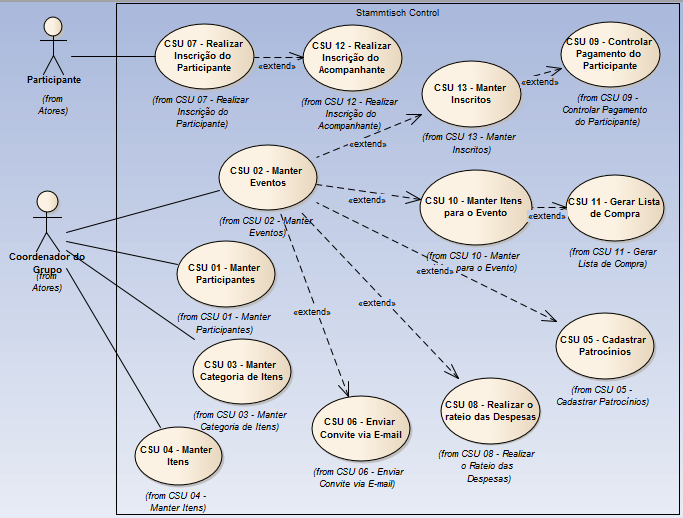
### Requisitos não-funcionais

A figura abaixo apresenta os requisitos não-funcionais de produto do projeto de desenvolvimento do sistema Stammtisch Control.



### Casos de Uso

A figura abaixo apresenta uma visão geral dos casos de uso do projeto de desenvolvimento do sistema Stammtisch Control.



## Realização dos testes

A lista abaixo relaciona as situações que foram identificadas como alvos de teste. Essa lista representa o que será testado nos casos de uso, requisitos funcionais e não-funcionais. Sendo que os cenários para testes destes itens serão apresentados no item 6 – Estrutura dos Testes.

Nem todos os alvos de teste serão realizados neste momento, sendo que algumas situações serão desenvolvidas em futuras iterações, conforme já delimitado no escopo.

### Teste Unitário

* Verifique que todas as classes ou métodos executam exatamente o processamento esperado.
* Verifique se os cálculos são realizados corretamente.

### Teste de Integração

* Verifique que os componentes do sistema produzem o resultado esperado quando integrados.
* Verifique se as telas de manutenção retornam os registros cadastrados no banco de dados.
* Verifique se as telas de alteração retornam somente os dados do registro que foi selecionado.

### Teste de Sistema e Interface do Usuário

* Navegue através de todos os casos de uso, verificando que cada tela de interface gráfica pode ser facilmente entendida e utilizada.
* Verifique que todas as palavras e expressões apresentadas nos diversos idiomas estão em conformidade com as devidas normas sintáticas e gramaticais.
* Verifique se os campos aceitam a informação que é proposta.
* Verifique que o sistema respeite as chamadas de telas e menus.
* Verifique se o sistema retorna mensagem de erro quando não informados campos obrigatórios.

### Teste do Banco de Dados

* Verifique que as informações do participante podem ser cadastradas, consultadas e removidas.
* Verifique que as informações de categorias de item, os itens, podem ser inseridos, atualizados e consultados.

### Teste Funcional

* Verifique que qualquer usuário pode acessar sua própria conta através de login e senha.
* Verifique que os relatórios estão corretos.

### Teste do Ciclo de Negócios

* Nenhum.

### Teste de Performance

* Verifique o tempo de resposta da rede de internet.
* Verifique o tempo de consulta/atualização das telas.

### Teste de Carga

* Verificar a resposta do sistema com 10 usuários.
* Verificar a resposta do sistema com 50 usuários.
* Verificar a resposta do sistema com 100 usuários.
* Verificar a resposta do sistema com 200 usuários.
* Verificar a resposta do sistema com 500 usuários.

### Teste de Stress

* Nenhum.

### Teste de Volume

* Nenhum.

### Teste de Segurança e de Controle de Acesso

* Verificar que usuários não cadastrados não podem acessar informações restritas aos cadastrados.
* Verificar que além do administrador, ninguém mais pode inserir, atualizar ou remover dados do sistema, exceto a inscrição do participante que pode ser feita somente por ele mesmo.

### Teste de Falha/Recuperação

* Nenhum.

### Teste de Instalação

* Verifique que a instalação do sistema ocorre normalmente no servidor.

# Premissas

Os testes devem ter uma cobertura de no mínimo 80% do código desenvolvido, o ideal seria 100%, porém como ocorrem situações em que é necessário alterar código fonte do sistema para conseguir testar, deixando o código muitas vezes ilegível, é preferível manter a legibilidade e testar manualmente tal funcionalidade.

O índice máximo de erros deve ser próximo a zero, considerando a existência dos testes unitários, de integração e de testes de sistema, todos automatizados, havendo a ocorrência de erros durante o processo de verificação devem ser corrigidos antes de liberar a versão para o cliente.

Porém, as áreas não abrangidas nos testes podem incorrer em erros, que assim que detectados devem ser corrigidos conforme considerada a sua severidade, entre: Alta, Média e Baixa, descritas no quadro a seguir.

|  |  |
| --- | --- |
| **Severidade** | **Descrição** |
| Alta | Bloqueia completamente a utilização de uma funcionalidade básica ou da aplicação inteira |
| Média | Bloqueia a utilização de uma funcionalidade. Mas, no entanto, a funcionalidade pode ser usada por meio da utilização de um contorno (*workaround*) conhecido |
| Baixa | Problemas cosméticos e solicitações de melhorias |

Define-se ainda que conforme a severidade, temos a prioridade de correção do referido erro detectado. Esta prioridade pode indicar que o erro deve ser corrigido conforme o Quadro abaixo.

|  |  |
| --- | --- |
| **Prioridade** | **Descrição** |
| 1 | O defeito deve ser corrigido imediatamente (até um dia útil). A aplicação não pode ser liberada sem a correção deste defeito |
| 2 | É altamente desejável que o defeito seja corrigido tão logo seja possível (até cinco dias úteis). A aplicação não pode ser liberada sem a correção deste defeito |
| 3 | Defeito de baixa prioridade. A aplicação pode ser liberada com esse defeito |

Os índices que se pretende chegar com este Plano de Testes são de liberar os *releases* do software sem nenhum erro ao cliente.

# Análise de riscos

Neste item são elencados os riscos que foram identificados que podem ocorrer durante o processo de implementação e execução deste plano de teste, bem como é apresentada uma contingência para cada uma das situações.

Alguns riscos envolvem situações não testadas nesta versão do plano de testes, porém já foram identificadas como possíveis problemas para a produção de uma nova versão mais completa e mais abrangente.

Os itens analisados foram classificados como: Ferramentas, quando envolvem as ferramentas de teste utilizadas; Execução, quando envolve o processo de execução dos cenários e testes automatizados; e Implementação, quanto envolvem o processo de desenvolvimento e codificação dos testes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Descrição** | **Classificação** | **Risco** | **Contingência** |
| Ferramenta de integração continua para rodar os testes automatizados | Ferramentas | Médio | Procurar ferramenta similar que suporte a execução dos testes automatizados e dispõe de algum mecanismo de aviso para a equipe quando algum teste falhar |
| Falta de ferramenta de testes de integração | Ferramentas | Crítico | Procurar ferramenta similar que atenda as necessidades do plano de testes. |
| Falta de ferramenta de testes de sistema | Ferramentas | Crítico | Procurar ferramenta similar que atenda as necessidades do plano de testes. |
| Falta de ferramenta de testes de stress | Ferramentas | Alto | Procurar ferramenta similar que atenda as necessidades do plano de testes. |
| Entrega tardia da aplicação para testes | Execução | Médio | Realização de horas-extras pela equipe de teste para cumprir o cronograma e/ou alocação de mais recursos humanos para execução dos testes. |
| Falta de conhecimento na ferramenta de execução testes | Execução | Médio | Promover um treinamento intensivo com a equipe de testes com foco total na ferramenta utilizada. |
| Mudança dos requisitos do projeto | Implementação | Médio | Propor implementar as mudanças e os casos de teste em outra entrega (iteração) com um novo cronograma, pois o escopo já foi aprovado pelo cliente. |
| Dificuldades com ambiente de aplicação durante os testes | Execução | Baixo | Acionar suporte da empresa especializada que foi contratada para prover o servidor de aplicação e banco de dados para manutenção. |
| Mudanças no cronograma de entrega do projeto | Implementação e Execução | Crítico | Realização de horas-extras pela equipe de teste para cumprir o cronograma e/ou alocação de mais recursos humanos para execução dos testes. |
| Quantidade excessiva de erros no sistema ocasionando retrabalho na execução dos testes | Execução | Alto | Estabelecer a execução de testes em paralelo com a alocação de mais terminais de teste e/ou mais recursos humanos. |

# Ambiente de Testes

Para a execução dos testes serão necessários recursos humanos, que serão detalhados no item 7.1 a seguir, que farão uso das ferramentas e hardwares especificados neste item para a execução do plano de teste.

Neste item serão também apresentados os principais tipos de teste com um breve detalhamento sobre cada tipo e a respectiva técnica utilizada.

Os profissionais que irão executar estes testes necessitam de conhecimentos básicos nas ferramentas que serão utilizadas, bem como necessitam conhecer as regras de negócio implementadas no Stammtisch Control, para um melhor entendimento devem conhecer o documento de Especificação de Requisitos, descrito anteriormente no item 1.4.

Os profissionais que irão desenvolver e implementar os testes, necessitam de conhecimentos na linguagem de programação utilizada para o desenvolvimento dos testes unitários, bem como nas ferramentas de criação dos testes de integração, de sistema e de stress.

Caso os membros da equipe selecionada desconheçam as ferramentas ou técnica utilizadas, será requerido um treinamento específico antes do início das atividades, nas ferramentas elencadas no item 5.2 conforme seu papel ou responsabilidade.

Os dados para realização dos testes poderão ser inseridos aleatoriamente conforme os campos presentes nas interfaces que serão testadas, não é requerido popular a base de dados com informações específicas, pois poderão ser feitas pela própria interface.

Para o desenvolvimento dos testes unitários e de integração, é necessário ter conhecimento de como criar cenários de testes com NUnit e Selenium.

* NUnit é um framework para criação de testes unitários e integração para .NET.
* Selenium é um framework para testes funcionais de interface web.

O desenvolvedor deve ter habilidades para escrever os cenários de testes especificados e percepção para desenvolver novos testes caso haja necessidade para manter a qualidade do sistema.

É importantíssimo ter conhecimento em TDD (Test Drive Development) para o desenvolvimento da aplicação, sendo deve ser criado primeiro o cenário de teste e posteriormente desenvolvido a funcionalidade necessária para fazer o teste passar.

Contudo também é requerido um bom conhecimento na linguagem de programação .NET e familiaridade com a IDE Visual Studio, utilizada no desenvolvimento deste projeto.

## Tipos de Teste

Neste item são apresentados os tipos de testes possíveis, seus objetivos, as técnicas utilizadas, os critérios de sucesso e as considerações especiais (se houver).

As transações abaixo se referem às “transações lógicas de negócio”. Essas transações são definidas como funções específicas que um usuário final do sistema é suposto de executar ao usar a aplicação, tais como adicionar ou modificar uma determinada informação.

### Teste Unitário

|  |  |
| --- | --- |
| **Objetivo do Teste:** | Garantir que todas as classes ou métodos estão funcionando adequadamente conforme foram projetadas e implementadas.  Testar a menor unidade possível do sistema. |
| **Técnica:** | Teste de caixa-preta.  Invocar as operações em cada classe e método informando como parâmetro valores pré-definidos, sendo aguardado um determinado resultado esperado. |
| **Critério de Finalização:** | Todos as classes e métodos retornem exatamente o resultado esperado. |
| **Considerações Especiais:** | O teste necessita de uma ferramenta específica para a linguagem de programação utilizada, onde serão implementadas classes de teste unitário que serão executadas em determinado tempo. |

### Teste de Integração

|  |  |
| --- | --- |
| **Objetivo do Teste:** | Verificar se os componentes do sistema, juntos, trabalham conforme descrito nas especificações do sistema e do projeto do programa. |
| **Técnica:** | Utilização de Mocks Objects, ou seja, classes que podem simular o comportamento de objetos que talvez não sejam acessíveis para testes.  Top-down, verificando primeiramente os componentes de alto nível do sistema até os itens que estão mais abaixo na hierarquia.  Bottom-up, que envolve integrar e testar os módulos de nível inferior na hierarquia e, então, subir na hierarquia de módulos, até que o módulo final seja testado. |
| **Critério de Finalização:** | Todas as relações entre os componentes integrados do sistema produzem o resultado esperado. |
| **Considerações Especiais:** | Algumas técnicas podem necessitar de uma ferramenta específica para a linguagem de programação utilizada. |

### Teste de Sistema e Interface do Usuário

|  |  |
| --- | --- |
| **Objetivo do Teste:** | Verificar o funcionamento do sistema com a integração dos componentes em um ambiente similar ao ambiente de operação do cliente.  Verificar ainda o seguinte:   * A navegação através dos alvos de teste reflete as funções e os requisitos do negócio apropriadamente, incluindo janela-a-janela, campo-a-campo, e o uso de métodos de acesso (tecla TAB, movimentos do mouse, teclas aceleradoras). * Objetos e características da janela, tais como menus, tamanho, posição, estado e foco conformam-se aos padrões. |
| **Técnica:** | Criar ou modificar os testes para cada janela para verificar a navegação e os estados de objeto apropriados para cada janela e objetos da aplicação.  Utilizar ferramentas para testar interfaces, onde podemos criar um fluxo de “cliques” ou entrada de dados para serem executados continuamente. |
| **Critério de Finalização:** | É verificado que cada janela permanece consistente com a versão de comparação ou dentro de padrões aceitáveis.  Que nenhum dos testes seja executado com erro. |
| **Considerações Especiais:** | Nem todas as propriedades para objetos personalizados e terceirizados podem ser acessadas. |

### Teste de Integridade de Dados e do Banco de Dados

|  |  |
| --- | --- |
| **Objetivo do Teste:** | Garantir que os métodos e processos de acesso ao banco de dados funcionam apropriadamente e sem corrupção dos dados. |
| **Técnica:** | Invocar cada método e processo de acesso ao banco de dados, alimentando cada um com dados ou requisições de dados válidos e inválidos.  Inspecionar o banco de dados para garantir que os dados foram inseridos como pretendido, que todos os eventos do banco de dados ocorreram apropriadamente, ou revisar os dados retornados para garantir que os dados corretos foram recuperados pelas razões corretas. |
| **Critério de Finalização:** | Todos os métodos e processos de acesso à base de dados funcionam como projetados e sem nenhuma corrupção de dados. |
| **Considerações Especiais:** | O teste pode necessitar de um ambiente de desenvolvimento ou drivers de SGBD para inserir ou modificar os dados diretamente nas bases de dados  Processos devem ser invocados manualmente.  Bases de dados pequenas ou minimizadas (número de registros limitados) devem ser usados para aumentar a visibilidade de eventos não-aceitáveis. |

### Teste de Função

|  |  |
| --- | --- |
| **Objetivo do Teste:** | Garantir a funcionalidade apropriada do alvo do teste, incluindo navegação, entrada de dados, processamento e recuperação. |
| **Técnica:** | Executar cada caso de uso, fluxo de caso de uso, usando dados válidos e inválidos, para verificar o seguinte:   * Os resultados esperados ocorrem quando dados válidos são usados * As mensagens de erro ou aviso apropriadas são exibidas quando dados inválidos são usados. * Cada regra de negócio é aplicada apropriadamente |
| **Critério de Finalização:** | * Todos os testes planejados foram executados. * Todos os defeitos identificados foram tratados. |
| **Considerações Especiais:** | Nenhum |

### Teste de Performance

|  |  |
| --- | --- |
| **Objetivo do Teste:** | Verificar que os comportamentos de performance para as transações designadas ou funções de negócio sob as seguintes condições:   * Carga de trabalho normal antecipada * Carga de trabalho no pior caso antecipada. |
| **Técnica:** | * Usar Procedimentos de Teste desenvolvidos para Teste da Função ou Ciclo de Negócio. * Modificar os arquivos de dados para aumentar o número de transações ou os scripts para aumentar o número de iterações que ocorre a cada transação. * Scripts devem ser rodados em uma máquina (melhor caso para comparar um único usuário, uma única transação) e ser repetidas com múltiplos clientes (virtual ou real, ver Considerações Especiais abaixo). |
| **Critério de Finalização:** | Único usuário ou transação: finalização com sucesso dos scripts de testes sem nenhuma falha e dentro da alocação de tempo por transação esperada ou requisitada.  Múltiplas transações ou usuários: finalização bem sucedida dos scripts de teste sem qualquer falha e dentro da alocação de tempo aceitável. |
| **Considerações Especiais:** | Um teste abrangente de performance inclui ter uma carga de trabalho no servidor.  Há vários métodos que podem ser usados para executar isso, incluindo:   * “Direcionar transações” diretamente para o servidor, usualmente na forma de chamadas SQL. * Criar carga de usuário “virtual” para simular muitos clientes, normalmente várias centenas. Ferramentas de Emulação de Terminal Remoto (RTE) são usadas para atingir essa carga. Essa técnica também pode ser usada para carregar uma rede com “tráfego”. * Usar múltiplos clientes físicos, cada um rodando scripts de teste para gerar uma carga no sistema.   O teste de performance deve ser executado em uma máquina dedicada ou em um tempo dedicado. Isso permite controle total e mensuração precisa.  As bases de dados usadas para o Teste de Performance devem ser do tamanho real ou proporcionalmente iguais. |

### Teste de Carga

|  |  |
| --- | --- |
| **Objetivo do Teste:** | Verifique o tempo de resposta para as transações designadas ou casos de negócio sob condições variantes de carga de trabalho. |
| **Técnica:** | Use testes desenvolvidos para o Teste do Ciclo de Negócio ou Função.  Modifique os arquivos de dados para aumentar o número de transações ou os testes para aumentar o número de vezes que cada transação ocorre.  Utilizar ferramentas de simulação de requisições para testar o quanto é suportado pelo servidor. |
| **Critério de Finalização:** | Múltiplas transações ou usuários: finalização bem sucedida dos testes sem qualquer falha e dentro da alocação de tempo aceitável. |
| **Considerações Especiais:** | O teste de carga deve ser executado em uma máquina dedicada ou em um tempo dedicado. Isso permite o controle total a e mensuração precisa.  As bases de dados usadas para os testes de carga devem ou ser do tamanho real ou igualmente dimensionadas. |

### Teste de Segurança e Controle de Acesso

|  |  |
| --- | --- |
| **Objetivo do Teste:** | Segurança do Nível de Aplicação: Verifique que um ator pode acessar apenas aquelas funções ou dados para os quais o seu tipo de usuário tem permissão.  Segurança do Nível de Sistema: Verifique que apenas aqueles atores com acesso ao sistema e aplicações têm permissão de acessá-los. |
| **Técnica:** | Segurança do Nível de Aplicação: Identifique e liste cada tipo de usuário e as funções ou dados para os quais cada tipo tem permissão.  Crie testes para cada tipo de usuário e verifique cada permissão criando transações específicos para cada tipo de usuário.  Modifique o tipo de usuário e repita os testes para os mesmos usuários. Em cada caso, verifique que funções ou dados adicionais estão corretamente disponíveis ou negados.  Acesso de Nível de Sistema: deve ser revisado ou discutido com o administrador de rede ou de sistema apropriado. |
| **Critério de Finalização:** | Para cada tipo de ator conhecido as funções ou dados apropriados estão disponíveis, e todas as transações funcionam como esperado e rodam nos Testes de Função anteriores. |
| **Considerações Especiais:** | Quanto ao teste de acesso ao sistema, esse teste pode não ser necessário já que ele pode ser uma função da administração da rede ou sistema. |

### Teste de Instalação

|  |  |
| --- | --- |
| **Objetivo do Teste:** | Verifique que os alvos de teste instalam apropriadamente em cada configuração de hardware necessária sobre as seguintes condições:   * Uma nova instalação, em um nova máquina, que nunca fora anteriormente instalada com o Stammtisch Control. * Atualização, numa máquina onde o Stammtisch Control já fora previamente instalado, para a mesma versão * Atualização, numa máquina que já disponha do Stammtisch Control instalado, de uma versão mais velha |
| **Técnica:** | Manualmente ou desenvolvida por scripts automatizados, para validar a condição da máquina alvo – novo Stammtisch Control nunca instalado; Stammtisch Control na mesma versão ou versão mais velha já instalada.  Começar ou executar a instalação.  Usando um subgrupo predeterminado de scripts de teste de funções, rode as transações. |
| **Critério de Finalização:** | As transações do Stammtisch Control executam de forma bem sucedida, sem falha. |
| **Considerações Especiais:** | O teste de atualização em relação as diferentes versões do Stammtisch Control será possível somente futuramente, quando houverem novas releases. |

## Ferramentas de teste

Neste item serão apresentadas as ferramentas ou softwares que serão empregadas para o projeto do Stammtisch Control.

|  |  |
| --- | --- |
| **Atividade** | **Ferramenta** |
| Projeto do Software | Enterprise Architect |
| Projeto de Teste | Microsoft Word |
| Ferramentas do SGBD | PostgreSQL 9.3 |
| Desenvolvimento/Codificação | ASP.NET |
| Execução de Teste Unitário | Visual Studio (NUnit) |
| Execução de Teste de Integração | Visual Studio (NUnit) |
| Execução de Teste de Sistema | Visual Studio (Selenium) e navegador Chrome |
| Execução de Teste de Stress | Visual Studio |

A principal ferramenta no processo de execução deste plano de teste se restringe ao Visual Studio 2015, que é empregado tanto para codificação do sistema, quanto para a execução dos testes unitários.

A tabela seguinte expõe os recursos de sistema e hardware necessários para a execução do plano de testes.

|  |
| --- |
| **Recursos de Sistemas e Hardware** |
| Servidor de Banco de Dados/ Aplicação Web   * 1 computador com SQL Server 2014 e Windows Server 2012 |
| Ambiente de Desenvolvimento   * 4 computadores conectados a internet, com SQL Server local, IDE de desenvolvimento (Visual Studio 2015) e ferramentas de teste instaladas |
| Terminais Clientes   * 4 computadores conectados a internet com as ferramentas de testes de integração e testes de sistema instalados e sistema operacional Windows 8.1 ou superior |

O ambiente de testes, no que tange ao servidor da aplicação web e de banco de dados, serão instalados e configurados por empresa especializada e contratada para esta finalidade, provendo o serviço na nuvem, dispensando aquisição de equipamentos deste tipo na empresa.

Quanto aos terminais clientes poderão ser utilizados qualquer computador ou notebook com acesso a internet e com as ferramentas listadas instaladas.

# Estrutura dos testes

Neste item serão descritos os cenários de teste utilizados para o desenvolvimento e realização dos testes unitários, de integração e de sistema do Stammtisch Control.

Os demais tipos de testes, como os testes de performance (carga e stress), serão desenvolvidos e realizados em outra iteração de desenvolvimento.

Para cada item é descrito o nível de prioridade definido, podendo ser Prioridade: Alta, Média, Baixa. Este nível é definido conforme os riscos que cada rotina apresenta, do cronograma de trabalho e da disponibilidade de pessoal e será utilizado para decisão quando alguns cenários de teste precisarão ser selecionados para execução em detrimento de outros.

## Testes Unitários

|  |  |
| --- | --- |
| 1. Repositório: Salvar Informações | |
| **Prioridade: Alta** | |
| **Nome do Teste:** lancando\_excecao\_quando\_nao\_conseguir\_salvar | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Instanciar entidade 2. Chamar operação salvar do repositório 3. Repositório não consegue salvar no banco por falha de comunicação ou qualquer outro motivo.   Resultado esperado: Deve lançar exceção com mensagem condizente quando não conseguir persistir entidade no banco.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Cadastro de Item: Validar lista das Categorias | |
| **Prioridade: Média** | |
| **Nome do Teste:** obtendo\_tipos\_de\_rateio | |
| **Descrição** | Pré-condições: Possuir Categorias cadastradas.  Cenário:   1. Obter tipos de rateios da enum 2. Retornar lista contendo a descrição do valor da enum e valor numérico.   Resultado esperado: buscar e validar corretamente a lista de tipos de rateio com a descrição e valor.  Pós-condições: Nenhuma. |

## Testes de Integração

|  |  |
| --- | --- |
| 1. Repositório: Salvando Participante | |
| **Prioridade: Alta** | |
| **Nome do Teste:** salvando\_participante | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Instanciar participante informando todos os dados. 2. Salvar participante. 3. Recuperar participante do banco.   Resultado esperado: ao salvar deve persistir o participante no banco de dados.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Repositório: Buscando Participante para Alteração | |
| **Prioridade: Média** | |
| **Nome do Teste:** buscando\_participante | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Instanciar participante. 2. Salvar participante. 3. Buscar participante informando o id do participante salvo   Resultado esperado: ao buscar participante pelo identificador (Id) deve listar o participante correspondente e seus respectivos dados cadastrados.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Repositório: Exclusão de Participante | |
| **Prioridade: Baixa** | |
| **Nome do Teste:** excluindo\_participante | |
| **Descrição** | Pré-condições: Nenhuma  Cenário:   1. Instanciar participante 2. Salvar participante no banco 3. Excluir participante informado o id do participante salvo. 4. Buscar participante informando o id do participante excluído.   Resultado esperado: deve excluir somente o participante correspondente. Ao buscar participante pelo Id excluído deve retornar null.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Repositório: Alteração de Participante | |
| **Prioridade: Média** | |
| **Nome do Teste:** atualizando\_participante | |
| **Descrição** | Pré-condições: Nenhuma  Cenário:   1. Instanciar participante. 2. Salvar participante no banco. 3. Buscar participante do banco. 4. Alterar informação do participante 5. Salvar participante com informação alterada. 6. Buscar participante do banco.   Resultado esperado: deve persistir a informação alterada no banco.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Repositório: Buscar todos os Participantes | |
| **Prioridade: Média** | |
| **Nome do Teste:** obtendo\_todos\_os\_participantes | |
| **Descrição** | Pré-condições: Banco de dados estar sem nenhum registro  Cenário:   1. Instanciar dois participantes. 2. Salvar os dois participantes. 3. Obter todos os participantes.   Resultado esperado: ao obter todos os participantes deve listar os dois registros contidos no banco.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Repositório: Salvar Categoria | |
| **Prioridade: Alta** | |
| **Nome do Teste:** salvando\_categoria\_item | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Instanciar Categoria Item 2. Salvar categoria item 3. Buscar categoria item   Resultado esperado: deve persistir todas as informações de categoria item no banco ao salvar um novo registro.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Participante Controller: Listar todos os Participantes | |
| **Prioridade: Baixa** | |
| **Nome do Teste:** carregando\_view\_index\_com\_todos\_os\_participantes\_ordenados\_pelo\_nome | |
| **Descrição** | Pré-condições: Banco de dados estar sem nenhum registro.  Cenário:   1. Instanciar três participantes com nomes desordenados. 2. Salvar participantes 3. Chamar operação Index do Controller   Resultado esperado: deve redirecionar para a view Index com todos os participantes contidos no banco, ordenado pelo nome para apresentar ao usuário.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Participante Controller: Salvar Participante | |
| **Prioridade: Alta** | |
| **Nome do Teste:** salvando\_participante | |
| **Descrição** | Pré-condições: Banco de dados estar sem nenhum registro.  Cenário:   1. Instanciar participante 2. Chamar operação FrmCadastroParticipante informando o participante instanciado 3. Buscar todos os participantes do banco.   Resultado esperado: deve conter somente um registro no banco, somente o respectivo participante.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Participante Controller: Retornar para Lista de Todos os Participantes | |
| **Prioridade: Baixa** | |
| **Nome do Teste:** retornando\_view\_index\_com\_os\_participantes\_ao\_salvar\_participante | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Instanciar dois participantes 2. Salvar participantes no banco 3. Instanciar um participante. 4. Chamar operação FrmCadastrarParticipante informando o participante   Resultado esperado: após salvar o participante deve redirecionar para a view index com todos os participantes para apresentar ao usuário.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Participante Controller: Cadastro de Participante | |
| **Prioridade: Alta** | |
| **Nome do Teste:** retornando\_view\_para\_cadastro\_de\_participante | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Chamar operação FrmCadastroParticipante.   Resultado esperado: deve redirecionar para a view de cadastro de participante.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Participante Controller: Alterar Participante | |
| **Prioridade: Média** | |
| **Nome do Teste:** carregando\_participante\_para\_editar | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Instanciar participante 2. Salvar participante 3. Chamar operação FrmEditarParticipante informando o Id do participante salvo   Resultado esperado: carregar participante e redirecionar para a tela de edição do registro.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. Participante Controller: Salvando alteração do Participante | |
| **Prioridade: Média** | |
| **Nome do Teste:** salvando\_edicao\_do\_participante | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Instanciar participante 2. Salvar participante. 3. Buscar participante 4. Alterar informação do participante 5. Chamar operação FrmEditarParticipante informando o Id e o participante alterado   Resultado esperado: deve persistir informações alteradas no banco.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. CategoriaItem Controller: Listar todas as Categorias | |
| **Prioridade: Média** | |
| **Nome do Teste:** carregando\_view\_index\_com\_todas\_categorias\_item\_ordenadas\_pela\_descricao | |
| **Descrição** | Pré-condições: Banco de dados estar sem nenhum registro.  Cenário:   1. Instanciar três categorias item com descrição desordenada. 2. Salvar categorias item. 3. Chamar operação Index.   Resultado esperado: deve redirecionar para a view Index com todos os registros de Categorias contidos no banco de dados, ordenando pela descrição para apresentar ao usuário.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. CategoriaItem Controller: Inclusão de Categorias | |
| **Prioridade: Alta** | |
| **Nome do Teste:** retornando\_view\_para\_cadastro\_de\_categoria\_item | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Chamar operação FrmCadastroCategoriaItem.   Resultado esperado: deve redirecionar para a tela de cadastro de categoria de item trazendo os campos para preenchimento.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. CategoriaItem Controller: Salvando Categorias | |
| **Prioridade: Alta** | |
| **Nome do Teste:** salvando\_categoria\_item | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Instanciar categoria item 2. Chamar operação FrmCadastroCategoriaItem informando a categoria item.   Resultado esperado: deve persistir categoria item no banco quando submetido requisição.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| 1. CategoriaItem Controller: Retorna visualização de Categoria após salvar | |
| **Prioridade: Baixa** | |
| **Nome do Teste:** retornando\_view\_index\_com\_as\_categorias\_item\_ao\_salvar\_categoria\_item | |
| **Descrição** | Pré-condições: Banco de dados estar sem nenhum registro..  Cenário:   1. Instanciar duas categorias item. 2. Salvar categorias no banco 3. Instanciar uma categoria item. 4. Chamar operação FrmCadastrarCategoriaItem informando a categoria   Resultado esperado: após salvar categoria item deve redirecionar para a view Index contendo todos os registros de categoria item do banco.  Pós-condições: Nenhuma. |

## Testes de Sistema

|  |  |
| --- | --- |
| **1. Cadastro de Participantes: Acesso à tela** | |
| **Prioridade: Alta** | |
| **Nome do Teste:** acessando\_tela\_de\_cadastro\_de\_participante | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Usuário acessa a tela de cadastro de Participante e clica na ação de Inclusão 2. Sistema exibe os campos da tela.   Resultado esperado: Ao clicar no botão novo deve abrir a tela de cadastro de participante.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| **2. Cadastro de Participantes: Inclusão** | |
| **Prioridade: Alta** | |
| **Nome do Teste:** cadastrando\_participante | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Usuário acessa a tela de cadastro de Participante e clica na ação de Inclusão 2. Sistema exibe os campos da tela. 3. Usuário informa todos os dados da tela e clica em Salvar. 4. Sistema valida os dados e Grava as informações no banco de dados.   Resultado esperado: ao informar os dados e clicar em salvar, deve salvar um registro no banco de dados.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| **3. Cadastro de Participantes: Mensagem de Campos Obrigatórios** | |
| **Prioridade: Média** | |
| **Nome do Teste:** apresentando\_mensagem\_de\_campos\_obrigatorios | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Usuário acessa a tela de cadastro de Participante e clica na ação de Inclusão 2. Sistema exibe os campos da tela. 3. Usuário NÃO informa os dados obrigatórios da tela e clica em Salvar. 4. Sistema valida os dados e Exibe mensagem de erro.   Resultado esperado: ao clicar em salvar sem informar os campos obrigatórios deve apresentar mensagem para cada campo informando que é obrigatório o preenchimento.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| **4. Cadastro de Participantes: Mensagem de E-mail Inválido** | |
| **Prioridade: Média** | |
| **Nome do Teste:** apresentando\_mensagem\_de\_email\_invalido | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Usuário acessa a tela de cadastro de Participante e clica na ação de Inclusão 2. Sistema exibe os campos da tela. 3. Usuário informa todos os dados da tela, porém informa um e-mail inválido (sem o @, por exemplo) e clica em Salvar. 4. Sistema valida os dados, Exibe mensagem de erro e não permite Salvar.   Resultado esperado: ao informar e-mail inválido deve ser apresentada uma mensagem para o usuário e não permitir salvar.  Pós-condições: Usuário corrige o e-mail e clica em Salvar. |

|  |  |
| --- | --- |
| **5. Cadastro de Participantes: Validação do CPF** | |
| **Prioridade: Média** | |
| **Nome do Teste:** apresentando\_mensagem\_da\_quantidade\_de\_digitos\_do\_cpf | |
| **Descrição** | Pré-condições: Nenhuma.  Cenário:   1. Usuário acessa a tela de cadastro de Participante e clica na ação de Inclusão. 2. Sistema exibe os campos da tela. 3. Usuário informa todos os dados da tela, porém informa um CPF inválido (com 9 dígitos, por exemplo) e clica em Salvar. 4. Sistema valida os dados, Exibe mensagem de erro e não permite Salvar.   Resultado esperado: quando informado CPF com a quantidade de dígitos diferente de 11 deve apresentar mensagem que o CPF deve conter 11 caracteres.  Pós-condições: Usuário corrige o CPF e clica em Salvar. |

|  |  |
| --- | --- |
| **6. Alteração de Participantes: Acesso à tela** | |
| **Prioridade: Alta** | |
| **Nome do Teste:** acessando\_tela\_de\_edicao\_do\_participante | |
| **Descrição** | Pré-condições: Possuir Participantes cadastrados.  Cenário:   1. Usuário acessa a tela de cadastro de Participante. 2. Sistema exibe o botão de Inclusão e Lista os registros existentes. 3. Usuário clica sobre algum registro já existente na tela. 4. Sistema abre a tela de edição do participante com os dados existentes já preenchidos.   Resultado esperado: ao clicar sobre o nome do participante deve abrir tela para edição dos dados.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| **7. Alteração de Participantes: Atualizar Informações** | |
| **Prioridade: Alta** | |
| **Nome do Teste:** atualizando\_informacoes\_do\_participante | |
| **Descrição** | Pré-condições: Possuir Participantes cadastrados.  Cenário:   1. Usuário acessa a tela de cadastro de Participante. 2. Sistema exibe o botão de Inclusão e Lista os registros existentes. 3. Usuário clica sobre algum registro já existente na tela. 4. Sistema abre a tela de edição do participante com os dados existentes já preenchidos. 5. Usuário altera os dados do participante e clica em salvar. 6. Sistema valida os dados e Grava as informações no banco de dados.   Resultado esperado: ao alterar informações do participante e clicar em salvar deve persistir no banco.  Pós-condições: Nenhuma. |

|  |  |
| --- | --- |
| **8. Alteração de Participantes: Mensagem de Campos Obrigatórios** | |
| **Prioridade: Média** | |
| **Nome do Teste:** apresentando\_mensagem\_de\_campos\_obrigatorios | |
| **Descrição** | Pré-condições: Possuir Participantes cadastrados.  Cenário:   1. Usuário acessa a tela de cadastro de Participante. 2. Sistema exibe o botão de Inclusão e Lista os registros existentes. 3. Usuário clica sobre algum registro já existente na tela. 4. Sistema abre a tela de edição do participante com os dados existentes já preenchidos. 5. Usuário apaga as informações existentes nos campos obrigatórios e Clica em salvar. 6. Sistema valida os dados, Exibe mensagem de erro informando sobre os campos obrigatórios e Não permite salvar os dados.   Resultado esperado: ao tentar salvar com alguma informação em branco deve apresentar mensagem de campos obrigatórios.  Pós-condições: Usuário preenche os dados novamente e clica em Salvar. |

# Recursos Humanos e Cronograma

Essa seção apresenta os recursos recomendados para a execução dos testes no projeto de desenvolvimento do sistema Stammtisch Control, bem como o cronograma de execução deste plano de testes.

No que tange aos Recursos Humanos serão apresentadas as funções, suas principais responsabilidades e seus conhecimentos ou conjunto de habilidades.

## Funções e Responsabilidades

Essa tabela mostra as suposições de recrutamento para o projeto, ou seja, as pessoas envolvidas nos testes e as suas responsabilidades. Constam na tabela a formulação de uma equipe completa de testes

|  |  |  |
| --- | --- | --- |
| **Recursos Humanos** | | |
| Função | Recursos Mínimos | Responsabilidades Específicas ou Comentários |
| Gerente de Teste,  Gerente do Projeto de Teste | Uma pessoa | Fornece supervisão gerencial.  Responsabilidades:   * provê direcionamento técnico * adquire recursos apropriados * fornece relatórios de gerenciamento |
| Test Designer | Uma pessoa | Identifica, prioriza, e implementa os casos de teste.  Responsabilidades:   * gera o plano de teste * cria o modelo de teste * avalia a efetividade do esforço de teste |
| Testador | Uma pessoa | Executa os testes.  Responsabilidades:   * executar os testes * registrar os resultados * reestabelecer-se dos erros * documentar solicitações de mudança |
| Administrador do Sistema de Teste | Uma pessoa | Garante que o ambiente e os bens de teste sejam gerenciados e mantidos.  Responsabilidades:   * administrar o sistema de gerenciamento teste * instalar e gerenciar o acesso do trabalhador ao sistema de testes |
| Gerente do Banco de Dados,  Administrador do Banco de Dados | Uma pessoa | Garante que o ambiente e bens de teste de dados (banco de dados) sejam gerenciados e mantidos.  Responsabilidades:   * administrar os dados de teste (base de dados) |
| Designer | Uma pessoa | Identifica e define as operações, atributos, e associações das classes de teste.  Responsabilidades:   * identificar e definir as classes de teste * identificar e definir os pacotes de teste |
| Implementador | Uma pessoa | Implementa e faz os testes unitários das classes e pacotes de teste.  Responsabilidades:   * cria as classes e pacotes de teste implementados no modelo de teste |

## Cronograma

|  |  |  |
| --- | --- | --- |
| **Atividades** | **Data de Início** | **Data de Término** |
| Planejar Teste | 22/02/2016 | 25/02/2016 |
| Projetar Teste | 26/02/2016 | 27/02/2016 |
| Implementar Teste | 27/02/2016 | 05/03/2016 |
| Executar Teste | 06/03/2016 | 10/03/2016 |
| Avaliar Teste | 11/03/2016 | 12/03/2016 |

# Gerência de defeitos

O principal objetivo do teste de software é medir o nível de qualidade de um sistema, seja a aplicação executável ou os artefatos utilizados na sua construção. A qualidade de um sistema pode ser medida, essencialmente, pelo número de falhas encontradas durante a execução dos testes.

Falha, nesse contexto, é a consequência de um erro, defeito ou engano. Ou seja, é o desvio entre o que foi solicitado pelo usuário por meio dos requisitos e o comportamento apresentado pela aplicação executável. Em virtude da complexidade e tamanho de um sistema ou para atender normas de qualidade ou processos de maturidade, se faz necessário utilizar um processo de gestão de defeitos integrado ao ciclo de vida de desenvolvimento e teste.

Um processo de gestão de defeitos tem o objetivo de definir práticas para prevenir os defeitos e minimizar os riscos de um projeto. A utilização de uma ferramenta automatizada, além de oferecer uma base comum para a entrada de informações, também oferece um meio para fomentar a integração entre o time de desenvolvimento e o time de testes. Além disso, por meio dos relatórios de gestão e métricas geradas por essas ferramentas, os gestores do projeto poderão promover a melhoria contínua do processo estabelecido.

## Métricas de Software para Testes

Uma métrica é um indicador de qualidade. Através da sua utilização, questões relacionadas ao software poderão ser respondidas. É uma excelente forma de verificar se os objetivos traçados para o processo de testes estão sendo alcançados, desde os pequenos projetos até mesmo nos casos em que o projeto é extenso e complexo. Através das métricas de teste, é possível traduzir a visão do negócio.

Ao escolher uma métrica, três fatores devem ser priorizados: a simplicidade, a facilidade na coleta e a relevância que terá dentro do seu processo de teste.

Algumas métricas podem ser coletadas na execução destes casos de teste de forma simples, por exemplo:

* **Base histórica**: existe um registro histórico contendo informações detalhadas de projetos passados? Através desses dados é possível mensurar, para os próximos projetos a complexidade, a qualidade da especificação de requisitos e a experiência da equipe?
* **Número de Ocorrências**: das ocorrências encontradas quantas são melhoria, quantas são efetivamente um erro e quantas não passam de sugestão?
* **Gravidade dos Defeitos**: qual a importância do bug encontrado? Ou melhor, qual o impacto pode sofrer o negócio em relação ao bug encontrado? É trivial, média ou obstáculo?
* **Tempo Médio para Encontrar um Defeito**: quais foram os esforços necessários para encontrar o bug?
* **Efetividade de Caso de Teste**: os casos de teste estão encontrando defeitos efetivamente?
* **Número de Casos de Teste**: qual a quantidade de casos de teste criados, executados, passaram, falharam e foram bloqueados? Qual o tempo de execução da baseline?

Para este projeto de testes foram selecionadas duas métricas a serem empregadas:

* **Efetividade de Caso de Teste**: registrando se os casos de teste estão encontrando defeitos efetivamente.
* **Número de Casos de Teste**: registrando a quantidade de casos de teste criados, executados com sucesso e executados com falha.

Realizando desde o inicio da execução dos casos de teste os devidos registros, poderemos realizar comparações estatísticas e acompanhar a evolução do sistema e de seus casos e cenários de teste.

Serão utilizados os formulários a seguir para o registro das informações dos casos e cenários de teste. Para o registro quanto a efetividade dos Casos de Teste:

|  |  |
| --- | --- |
| **Registro de bug encontrado na execução de Caso de Teste** | |
| Nome do Caso de Teste: *Conforme cenários especificados.* | |
| Data: \_\_\_/\_\_\_/\_\_\_\_\_\_ | Hora: \_\_\_:\_\_\_ |
| Reproduzível: 🞏 Sim 🞏 Não | Reincidente: 🞏 Sim 🞏 Não |
| Severidade:🞏 Alta 🞏 Média 🞏 Baixa | Prioridade: 🞏 Alta 🞏 Média 🞏 Baixa |
| Categoria do Defeito: *unitário, integração, banco de dados, sistema, validação, etc.* |  |
| Circunstância: *Relato do que estava sendo realizado, ou qual classe estava sendo alterada.* | |
| Dados utilizados no teste: *exemplos para reprodução.* | |

Para registro dos números de execução e criação dos casos de teste será utilizada uma planilha contendo pelo menos as seguintes informações: nome do cada caso de teste existente, data e hora de cada execução, o tempo que levou efetivamente e se a execução resultou em sucesso ou falha, conforme exemplo a seguir:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Controle dos Casos de Teste** | | | | |
| Caso de Teste | Data/Hora da execução | Tempo | Sucesso | Falha |
| Alteração de Participantes: Acesso à tela | 11/03/2016 14:40h | 00:02:00 | ◼ | 🞏 |
| 12/03/2016 15:35h | 00:02:07 | 🞏 | ◼ |
| 13/03/2016 10:32h | 00:01:49 | ◼ | 🞏 |
| Repositório: Salvar Categoria | 11/03/2016 14:49h | 00:02:05 | ◼ | 🞏 |
| 12/03/2016 16:35h | 00:01:50 | ◼ | 🞏 |
| ........ | ........ | ........ | ........ |

Os registros de todos os casos de teste podem auxiliar na tomada de decisão quando a ampliação dos cenários e dos casos de teste, pois podem comprovar sua importância no processo de desenvolvimento do sistema Stammtisch Control.

# Comunicação

Além das situações demonstradas no item anterior sobre as métricas e suas formas de registro, os testes unitários possuem o seu registro na própria ferramenta, exibindo uma lista de testes implementados e a quantidade de falhas ocorridas, porém sendo possível devem ser centralizadas em um controle único para facilitar a análise e tomada de decisão.

Durante a execução das tarefas de teste, ou seja, na realização dos cenários de teste, o(s) testador(es) farão o registro das atividades conforme formulários exemplificados no item 8.1 em uma planilha on-line específica.

Neste formulário estão listados todos os cenários e podem ser incluídos ainda o tempo de execução de cada um deles, sendo gerado um total concluído em percentual e um gráfico de conclusão. Poderá ser ainda implementado um gráfico de *burn-down* para uma visão mais precisa de tempo *versus* execução das tarefas.

Para reportar os defeitos encontrados na execução dos testes também serão utilizados formulários personalizados, contendo pelo menos as seguintes informações:

* Categoria do defeito (unitário, integração, banco de dados, sistema, validação, algoritmo, acessos, etc.)
* Severidade (Alta, Média ou Baixa) ou classificada por número.
* Prioridade (Alta, Media ou Baixa) ou classificada por número.
* Descrição do defeito ou do comportamento resultado ou a própria mensagem de erro.
* Reproduzível?
* Reincidente?
* Dados usados no teste
* Opcional: imagens que ilustrem o erro ou a situação da interface no momento do teste.

Com estes dados coletados e tabuados, a equipe de desenvolvimento da aplicação poderá simular a situação e prover a correção para uma nova bateria de testes.

Por fim, com todo o exposto almeja-se que o **Stammtisch Control** possa satisfazer as necessidades do cliente de forma completa, sem incorrer em erros e falhas, com o intuito de não prejudicar o andamento das atividades de preparação dos eventos e também de controle posterior dos rateios das despesas e pagamentos realizados.