Semana de Atualização em Tecnologia da Informação

Plano de Teste

Versão <1.0>

Histórico da Revisão

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autores** |
| <06/dezembro/15> | <1.0> | Primeira Versão | Dion Duarte  Lucas Fernandes |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Índice Analítico

[1. Introdução 3](#_Toc437176980)

[1.1 Finalidade 3](#_Toc437176981)

[1.2 Escopo 4](#_Toc437176982)

[1.3 Público-alvo 4](#_Toc437176983)

[1.4 Terminologia e Acrônimos do Documento 4](#_Toc437176984)

[1.5 Referências 4](#_Toc437176985)

[1.6 Estrutura do Documento 4](#_Toc437176986)

[2. Missão de Avaliação e Motivação dos Testes 5](#_Toc437176987)

[2.1 Informações Detalhadas 5](#_Toc437176988)

[2.2 Missão de Avaliação 5](#_Toc437176989)

[2.3 Motivadores dos Testes 5](#_Toc437176990)

[3. Itens-alvo dos Testes 5](#_Toc437176991)

[4. Resumo dos Testes Planejados 5](#_Toc437176992)

[4.1 Resumo das Inclusões dos Testes 6](#_Toc437176993)

[4.2 Resumo dos Outros Candidatos a Possível Inclusão 6](#_Toc437176994)

[4.3 Resumo das Inclusões dos Testes 6](#_Toc437176995)

[5. Abordagem dos Testes 6](#_Toc437176996)

[5.1 Técnicas e Tipos de Teste 6](#_Toc437176997)

[5.1.1 Teste de Unidade, Cadastro da Classe Edição 7](#_Toc437176998)

[5.1.2 Teste de Unidade, Cadastro da Classe Pessoa 10](#_Toc437176999)

[5.1.3 Teste de Unidade, Cadastro da Classe Dados Bancários 15](#_Toc437177000)

[5.1.4 Teste de Unidade, Cadastro da Programação 17](#_Toc437177001)

[5.1.5 Teste de Integração, Cadastro de Evento com Dados Bancários 19](#_Toc437177002)

[5.1.6 Teste de Integração, Cadastro de Edição com Programação 20](#_Toc437177003)

[5.1.7 Teste de Integração, Cadastro de Pessoa com Programação 20](#_Toc437177004)

# Introdução

## Finalidade

A finalidade do Plano de Teste de Iteração é reunir todas as informações necessárias ao planejamento e ao controle do esforço de teste referente a uma iteração específica. Ele descreve a abordagem dada ao teste do software e é o plano de nível superior gerado e usado pelos gerentes para coordenar o esforço de teste.

Este *Plano de Teste* referente a Semana de Atualização em Tecnologia da Informação suporta os seguintes objetivos:

* Devem ser verificadas as classes principais do sistema. O principal método testado será o cadastro, cujo qual determinará a consistência dos dados no sistema.
* Procurar realizar os testes para que não haja erros em outros elementos importantes do sistema, cujo os quais dependem da consistência dos dados recuperados do banco de dados.
* Procura-se avaliar o funcionamento do sistema com os testes de integração e de unidade. Será aplicado o teste de caixa branco e o caminho básico para analisar o comportamento das funções.
* Será necessário uma IDE Java apropriada e a ferramenta JUnit para testar e validar os requisitos descritos nesse documento.

## Escopo

Este documento descreve dois níveis de teste: Unidade e de Integração. No teste de unidade verifica a consistência dos dados antes de realizar o registro no banco de dados. Enquanto no teste de integração, o objetivo é verificar os relacionamentos entre os objetos e se mesmo após o registro, é possível recuperar tais informações.

As funções testadas neste documento se preocupam somente com o teste sobre as funcionalidades do sistema. deOutros itens de teste serão deixados para versões futuras do documento.

## Público-alvo

Este documento busca oferece uma explicação para os clientes do sistema SATI. Busca-se enaltecer as funcionalidades descritas no Product Backlog do sistema, quando foi realizado o levantamento de requisitos do sistema.

Procura-se manter a consistência dos dados e do cadastro correto no banco de dados para que não haja dados errados e que acarretam no mal desempenho do sistema.

## Terminologia e Acrônimos do Documento

* SATI – Semana de Atualização em Tecnologia da Informação.

## Referências

Especificação dos requisitos de software (APS1 Eng.Soft. 02/2015)

Estudo de caso da SATI (Semana de Atualização em Tecnologia da Informação), material da disciplina de engenharia de software, elaborado pelo prof. Mathias Talevi Betim. Disponível em: <<http://moodle.pg.utfpr.edu.br/mod/resource/view.php?id=28856>>. Acesso mediante usuário e senha da instituição. Disponível para o segundo semestre de 2015.

## Estrutura do Documento

Na segunda sessão deste documento, busca-se explanar sobre os itens tratados nessa versão do documento de plano de teste. Em seguida no item 3, comenta-se sobre os itens a serem testados. No item 4, é falado sobre o planejamento dos testes, e por fim na 5 sessão, os testes são elaborados.

# Missão de Avaliação e Motivação dos Testes

Estes testes tem a missão de encontrar possíveis falhas e eventuais problemas na implementação das funções principais da aplicação, se considera falha tudo aquilo que descrito no documento de requisitos não está sendo satisfeito, requisitos funcionais e não funcionais e requisitos de usuário.

## Informações Detalhadas

Este plano de teste pretende avaliar 4 classes, que tem comportamentos distintos, mas que envolvem o cerne da aplicação que são os cadastros, o CRUD para cada uma das classes é necessário e foco principal deste teste.

Para que o CRUD funcione da maneira correta os requisitos devem ser satisfeitos, tudo o que dentro da regra de negócio está sendo cuidadosamente tratado, deve ser verificado, para a verificação de cada classe sãos utilizados os testes unitários e para uma abordagem mais ampla os testes de integração.

Vale ressaltar que os testes de integração são realizados entre classes distintas, mas dependentes, e que os diferentes módulos do sistema foram escritos por desenvolvedores diferentes, além disso, garantir que empregando o método de desenvolvimento ágil os testes propostos neste documento sejam satisfeitos.

## Missão de Avaliação

* localizar o maior número de erros possíveis
* localizar problemas importantes entre as várias camadas do sistema.
* verificar uma especificação descritas nos documentos de requisitos.
* cumprir as determinações de cadastros
* A missão não é validar campos, está devera ser feita em tempo de *runtime* utilizando a camada de visualização fornecida pelos *forms* java, nestes testes essa camada não será utilizada.

## Motivadores dos Testes

* Riscos de qualidade, riscos técnicos, riscos do projeto, cumprir as histórias, requisitos funcionais, requisitos não funcionais, falhas ou erros suspeitos.

# Itens-alvo dos Testes

A listagem abaixo identifica os itens  software, hardware e elementos de suporte do produto  que foram identificados como alvo dos testes. Essa lista representa os itens que serão testados.

* Classe Edição
* Classe Dados Bancários
* Classe Programação
* Classe Pessoa
* Integração entre Edição e Dados Bancários
* Integração entre Programação e Edição
* Integração entre Programação e Pessoa

Não será realizado testes no hardware utilizado.

# Resumo dos Testes Planejados

* Os testes serão empregados nas classes de sistema, Edição, Dados Bancários, Pessoa e Programação.

## Resumo das Inclusões dos Testes

* Não serão incluídos testes de design e interação ou qualquer teste que tenha em foco a parte visual do sistema, ou validações de campos, pois estas são avaliadas em tempo de *runtime* que não chegariam até a regra de negócio, portanto não persistiriam.

## Resumo dos Outros Candidatos a Possível Inclusão

* No momento da elaboração deste plano não há algo que embora pensado não possa ser executado pela equipe.

## Resumo das Inclusões dos Testes

Alguns testes não precisam ser realizados, devido as restrições empregadas não regra de negócio e que não permitiram um estado inconsistente devolvendo ao programa o fluxo normal, exemplos são objetos que dependem da existência de outros e que não poderiam ser instanciados sem uma validação do objeto que o compõe.

* Testes de validação de campos não contribuem para alcançar a missão de avaliação, pois serão tratados em testes posteriores utilizando a interação do usuário.
* Testes de desempenho não serão realizados, pois não há recursos suficientes para executar esses testes.

# Abordagem dos Testes

A estratégia utilizada para analisar a técnica foi o Teste de Unidades através do método de Caixa Branca. Para descrever o algoritmo foi utilizado o método de teste de caminho básico.

# Técnicas e Tipos de Teste

## Teste de Unidade, Cadastro da Classe Edição

|  |  |
| --- | --- |
| Objetivo da Técnica: | Cadastrar uma edição do evento SATI |
| Técnica: | Definição das Variáveis:  Os dados de uma edição são: Título, Tema, Data Inicial, Data Final, Dados Bancários. Para a inserção de dados de uma edição no banco de dados, é necessário que os dados do objeto não sejam nulos ou vazios. Além disso, a data inicial do evento DEVE ser menor que a data final do evento e somente UMA edição pode se iniciar em um ano. O teste sobre Dados bancários será realizado na integração dos objetos.  As variáveis título e tema não possuem restrições adicionais.  Sobre a registrar de data inicial, não será possível registrar uma edição tal que:   * A data inicial seja 11-11-11 e uma data final 10-11-11. * A data inicial seja 10-11-11 e uma data final 10-11-11.   Ainda sobre a data inicial e final, é possível registrar uma edição tal que:   * A data inicial seja 09-11-11 e uma data final 10-11-11.   Se dois eventos que comecem no mesmo ano, apenas o primeiro será registrado, enquanto o registro do segundo é negado, por exemplo:   * Evento\_1: Data Inicial: 11-11-10 e Data Final 25-11-10. * Evento\_2: Data Inicial: 14-05-10 e Data Final 25-11-10.   Mesmo tendo uma data inicial menor, o Event\_2 tentou ser cadastrado depois de Evento\_1 e, portanto, não será inserido no banco de dados.  Porém quando,   * Evento\_3: Data Inicial: 11-11-10 e Data Final 25-11-10. * Evento\_4: Data Inicial: 14-05-09 e Data Final 25-11-10.   Mesmo terminando na mesma data, os eventos não são restringidos. A única restrição é na data inicial.  Caminho Básico:  A função *Cadastrar* a edição possui duas saídas possíveis, **verdadeiro** e **falso**. A primeira dá-se quando for possível inserir o objeto no banco de dados, logo a segunda é quando o mesmo não é possível.  O método recebe um objeto do tipo Edição contendo os dados do mesmo.  Primeiramente o método *Cadastrar* invoca uma função *VerificarAtributos* para verificar a consistência dos dados.  Dentro da função *VerificarAtributos*, verifica-se se os dados são nulos ou vazios, se algum for, a função retorna **falso**. Caso contrário, é verificado se a Data Inicial do evento é anterior a Data Final, se não, é retornado **falso**. Em seguida, é verificado o ano da edição, se já houver alguma outra edição iniciando-se naquele ano, é retornado **falso**. Se a função não tiver retornado em nenhum ponto desta verificação, é retornado o verdadeiro para a função *Cadastrar*.  Se *VerificarAtributos* é **falso**, a função retorna falso, avisando que não foi possível o cadastro.  Caso *VerificarAtributos* seja **verdadeiro**, então é instanciado uma chamada da camada de persistência e o objeto é enviado para a inserção no banco de dados.  Dentro da classe da persistência, se for possível a conexão com o banco, o dado será inserido com sucesso e retorna verdadeiro, caso não seja possível é retornado falso.  Por fim, a função *Cadastrar* irá retornar o mesmo valor retornado da persistência.  A complexidade ciclomática será 6, determinado pelo total de caminhos possíveis. |
| Estratégias: | A estratégia utilizada para analisar a técnica foi o Teste de Unidades através do método de Caixa Branca. Para descrever o algoritmo foi utilizado o método de teste de caminho básico. |
| Ferramentas Necessárias: | Computador com máquina virtual JAVA instalado.  Acesso à o banco de dados MySQL. |
| Critérios de Êxito: | A técnica suporta o teste de:   * Unidade, testando o método de cadastro de uma nova edição no banco de dados. * Caixa Branca, que foi utilizado para descrever o algoritmo utilizado.   Quando dados corretos forem enviados ao banco de dados, deverá ser possível realizar o cadastro. |
| Considerações Especiais: | Não Aplica. |

### 

## Teste de Unidade, Cadastro da Classe Pessoa

|  |  |
| --- | --- |
| Objetivo da Técnica: | * Cadastrar uma pessoa do evento SATI |
| Técnica: | Definição das Variáveis:  Os dados de uma pessoa são: Nome, Tipo, RA, E-mail, Instituição, CPF e RG. Para a inserção de dados de uma pessoa no banco de dados, é necessário que os dados do objeto não sejam nulos ou vazios, com exceção do seguinte caso:   * Se uma pessoa não possuir um RA, isto indica que a pessoa não possui ligação com a UTFPR, logo, é necessário informar a instituição a qual pertence. * Caso a pessoa possua RA, então a UTFPR é a instituição determinada para aquela pessoa.   Logo, se uma pessoa não possuir um RA, o campo Instituição deve ser preenchido, caso contrário, não será possível realizar o cadastro.  Se o RA for preenchido, automaticamente o sistema identifica a instituição como **UTFPR**, antes de cadastrar no banco de dados. Se o RA for preenchido e a Instituição também, o sistema ignora e preenche **UTFPR** como instituição e cadastra a pessoa no banco de dados.  O RA deve ser uma sequência de 7 números quaisquer.  Logo, o RA **1234567** é válido e poderá ser cadastrado e **ABC1!@24234** ou **12345678** não serão aceitos e não serão cadastrados.  O tipo da pessoa pode ser:   * Aluno * Professor * Servidor * Outros   Qualquer outra entrada para o campo Tipo é inválida. Note que não há diferenciação entre letras maiúsculas e minúsculas.  Logo, se o Tipo da pessoa for: *Aluno*, *Professor*, *Servidor* ou *Outros* o cadastro será realizado. Para **qualquer outro dado** informado, o cadastro será negado.  O e-mail da pessoa deve ser único, ou seja, somente um e-mail por pessoa cadastrada. Além disso, um e-mail válido deve obedecer a seguinte expressão regular:  *^[\\w\\.-]+@([\\w\\-]+\\.)+[A-Z]{2,4}$*  Logo, o email deve é composto por:   * Uma sequência de caracteres, exceto por ponto (.) e espaço vazio, * Seguido pelo símbolo arroba (@), * Depois novamente uma sequência de caracteres, exceto por ponto (.) e espaço vazio, * E por fim, de dois a quatro letras.   Sendo assim, o seguinte e-mail deve ser considerado válido:  **exemplo@email.sati**  Enquanto, **exemploNãoValido** não será cadastrado.  O CPF deve ser válido. Para um CPF ser válido, este deve ser composto por 11 números e seguir as regras do Governo Brasileiro para conferir se o CPF é válido.  O CPF válido **14661485785** será aceito, enquanto **12345678901** não será.  Nome e RG não possuem restrições adicionais.  Caminho Básico:  A função *Cadastrar* a pessoa possui duas saídas possíveis, **verdadeiro** e **falso**. A primeira dá-se quando for possível inserir o objeto no banco de dados, logo a segunda é quando o mesmo não é possível.  O método recebe um objeto do tipo Pessoa contendo os dados do mesmo.  Primeiramente o método *Cadastrar* invoca uma função *VerificarAtributos* para verificar a consistência dos dados.  Dentro da função *VerificarAtributos*, verifica-se se os dados são nulos ou vazios, se algum dado for, a função retorna **falso**. Caso contrário, é verificado se o e-mail é válido através da função *validarEmail.*  Dentro *validarEmail* de é verificado se o email segue o padrão descrito acima, se sim retorna **verdadeiro**, caso contrário **falso**.  Se o de *validarEmail* retorno é falso, a função *VerificarAtributos* retorna **falso**. Caso contrário segue a verificação do RA.  Se o RA for válido (definido anteriormente), segue-se a verificação dos dados, caso contrário, retorna **falso**.  Na sequência é testado o tipo da pessoa, se for *Aluno*, *Professor*, *Servidor* ou *Outros*, segue a para verificação do CPF, caso contrário retorna **falso**.  O CPF é testado pelo método *isCPF* da classe validar, caso seja válido, retorna **verdadeiro** para a função *Cadastrar*, caso contrário, retorna **falso**.  Se *VerificarAtributos* é **falso**, a função retorna falso, avisando que não foi possível o cadastro.  Caso *VerificarAtributos* seja **verdadeiro**, então é instanciado uma chamada da camada de persistência e o objeto é enviado para a inserção no banco de dados.  Dentro da classe da persistência, se for possível a conexão com o banco, o dado será inserido com sucesso e retorna verdadeiro, caso não seja possível é retornado falso.  Por fim, a função *Cadastrar* irá retornar o mesmo valor retornado da persistência.  A complexidade ciclomática será 8, determinado pelo total de caminhos possíveis. |
| Estratégias: | * A estratégia utilizada para analisar a técnica foi o Teste de Unidades através do método de Caixa Branca. Para descrever o algoritmo foi utilizado o método de teste de caminho básico. |
| Ferramentas Necessárias: | Computador com máquina virtual JAVA instalado.   * Acesso à o banco de dados MySQL. |
| Critérios de Êxito: | A técnica suporta o teste de:   * Unidade, testando o método de cadastro de uma nova edição no banco de dados. * Caixa Branca, que foi utilizado para descrever o algoritmo utilizado.   Quando dados corretos forem enviados ao banco de dados, deverá ser possível realizar o cadastro. |
| Considerações Especiais: | * Não Aplica. |

## Teste de Unidade, Cadastro da Classe Dados Bancários

|  |  |
| --- | --- |
| Objetivo da Técnica: | Cadastrar dados bancários do evento SATI |
| Técnica: | Definição das Variáveis:  Os dados de uma edição são: *Carteira, Agência, Conta* e *Tipo*. Para a inserção de dados dos dados bancários no banco de dados, é necessário que os dados *Agência, Conta* e *Tipo* o do objeto não sejam nulos ou vazios.  A restrição da Carteira é que a carteira seja maior que 0.  Logo, se Carteira for **1** ou **1000**, o objeto será cadastrado.  Caso a carteira seja **0** ou **-10**, o objeto não será cadatrado.  Caminho Básico:  A função *Cadastrar* os dados bancários possui duas saídas possíveis, **verdadeiro** e **falso**. A primeira dá-se quando for possível inserir o objeto no banco de dados, logo a segunda é quando o mesmo não é possível.  O método recebe um objeto do tipo Dados Bancários contendo os dados do mesmo.  Primeiramente o método *Cadastrar* invoca uma função *VerificarAtributos* para verificar a consistência dos dados.  Dentro da função *VerificarAtributos*, verifica-se se os dados são nulos ou vazios, se *Agência, Conta* ou *Tipo* forem, a função retorna **falso**. Caso contrário, é verificado se *Carteira* é maior ou igual a 0. Se for, é retornado **verdadeiro**, caso contrário retorna **falso**.  Se *VerificarAtributos* é **falso**, a função retorna falso, avisando que não foi possível o cadastro.  Caso *VerificarAtributos* seja **verdadeiro**, então é instanciado uma chamada da camada de persistência e o objeto é enviado para a inserção no banco de dados.  Dentro da classe da persistência, se for possível a conexão com o banco, o dado será inserido com sucesso e retorna verdadeiro, caso não seja possível é retornado falso.  Por fim, a função *Cadastrar* irá retornar o mesmo valor retornado da persistência.  A complexidade ciclomática será 5, determinado pelo total de caminhos possíveis. |
| Estratégias: | A estratégia utilizada para analisar a técnica foi o Teste de Unidades através do método de Caixa Branca. Para descrever o algoritmo foi utilizado o método de teste de caminho básico. |
| Ferramentas Necessárias: | Computador com máquina virtual JAVA instalado.  Acesso à o banco de dados MySQL. |
| Critérios de Êxito: | A técnica suporta o teste de:   * Unidade, testando o método de cadastro de uma nova edição no banco de dados. * Caixa Branca, que foi utilizado para descrever o algoritmo utilizado.   Quando dados corretos forem enviados ao banco de dados, deverá ser possível realizar o cadastro. |
| Considerações Especiais: | Não Aplica. |

## Teste de Unidade, Cadastro da Programação

|  |  |
| --- | --- |
| Objetivo da Técnica: | * Cadastrar uma programação no evento da SATI |
| Técnica: | * Os dados obrigatórios de uma programação são: * Datas de inicio e fim, as datas de programação devem estar dentro do prazo de inicio e fechamento da edição no ano em que se deseja efetuar o cadastro, o formato de uma data é completo, desta maneira quando mencionar data, neste documento, ela está composta da seguinte forma: YYYY-MM-DD HH:MM:SS; * Uma pessoa responsável, é representada por um objeto da classe pessoa; * Uma edição da qual a programação faz parte, é representado por um objeto da classe edição; * Um custo variável que pode ser alterado durante a edição; * Um custo fixo por responsável; * Local, título e descrição, descrição diz respeito ao tipo de programação exemplo, palestra, minicurso, etc; * Valor de inscrição.   A verificação dos campos ocorre em duas etapas, uma relacionada estritamente com os dados necessários a camada de persistência e outra que diz respeito as restrições do projeto.  Datas persistência – Os campos referentes a data e hora de inicio e data e hora de encerramento, são verificados, de maneira que a data inicial da programação não seja posterior a data de encerramento.  Local, datas e responsável. Apos a verificação de restrição entre as datas mencionadas, há também a verificação de restrição das datas junto a edição, que não permite a uma programação ser cadastrada numa data que está fora do período de inicio e fim da edição. Um responsável não pode estar ocupando dois lugares ao mesmo tempo, para essa verificação o sistema faz uma checagem de onde o responsável está na hora e local em que se deseja efetuar o cadastro. Para isso o sistema faz uma busca por data, sala e identificação do responsável.   * Caminho básico   A função *Cadastrar* a programação possui duas saídas possíveis, **verdadeiro** e **falso**. A primeira dá-se quando for possível inserir o objeto no banco de dados, logo a segunda é quando o mesmo não é possível.  O método recebe um objeto do tipo Programação contendo os dados do mesmo.Primeiramente o método *Cadastrar* invoca uma função *VerificarAtributos* para verificar a consistência dos dados. |
|  | Dentro da função *VerificarAtributos*, verifica-se se os dados são nulos ou vazios, se algum for, a função retorna **falso**. Caso contrário, é verificado se a Data Inicial da programação é anterior a Data Final, se não, é retornado **falso**. Em seguida, é verificado a data de inicio e fim que se deseja cadastrar a programação, se a edição já tem agenda definida, é retornado **falso**. A restrição de pessoa e responsável é mais complexa e é realizada por último, se o responsável está relacionado a outra programação em outro local entre o horário de inicio e fim desta programação a função retorna **falso**. Quando estes casos não forem satisfeitos a verificação retorna **verdadeiro**, sinalizando que os dados informados podem ser levados a persistência e a programação pode ser cadastrada. |
| Estratégias: |  |
| Ferramentas Necessárias: | * A técnica exige as seguintes ferramentas: * JUnit com ambiente de programação Java * Java Virtual Machine * Mysql com o schema definido |
| Critérios de Êxito: | A técnica suporta o teste de:   * Unidade, testando o método de cadastro de uma nova programação no banco de dados. * Caixa Branca, que foi utilizado para descrever o algoritmo utilizado.   Quando dados corretos forem enviados ao banco de dados, deverá ser possível realizar o cadastro. |
| Considerações Especiais: | Não se aplica. |

## Teste de Integração, Cadastro de Evento com Dados Bancários

|  |  |
| --- | --- |
| Objetivo da Técnica: | Cadastrar Evento com Dados bancários do evento SATI |
| Técnica: | Após um objeto da classe Dados Bancários passar pelo teste de unidade, é possível cadastrar uma edição associada a aquele objeto.  O objeto Evento pode ser cadastrado sem um objeto do tipo Dados Bancários, porém quando associado a tal, deve haver consistência entre os dados.  Quando cadastrar Evento com um objeto Dados Bancários, deve ser possível recuperar esses dados do banco de dados, ou seja, o ID do objeto Dados Bancários deve ser o mesmo do ID recuperado pela pesquisa da edição.  Logo, é possível adicionar uma edição com dados bancários e ao pesquisar tal edição, pode-se recuperar os dados da edição associada ao mesmo.  Deve ser possível cadastrar uma edição e verificar se a mesma está cadastrada no banco de dados.  Deve ser possível cadastrar os dados bancários e verificar se o mesmo está cadastrada no banco de dados.  Deve ser possível cadastrar uma edição com dados bancários associado e verificar se a mesma está cadastrada no banco de dados e os dados bancários também. |
| Estratégias: | A estratégia utilizada para analisar a técnica foi o Teste de Unidades através do método de Caixa Branca. Para descrever o algoritmo foi utilizado o método de teste de caminho básico. |
| Ferramentas Necessárias: | A técnica exige as seguintes ferramentas:   * JUnit com ambiente de programação Java * Java Virtual Machine * Mysql com o schema definido |
| Critérios de Êxito: | A técnica suporta o teste de:   * Integração, testando o método de cadastro de uma nova edição no banco de dados e a interação com dados bancários. * Caixa Branca, que foi utilizado para descrever o algoritmo e utilizado.   Quando dois objetos possuem dados corretos, deverá ser possível cadastrar o relacionamento entre classes no banco de dados. |
| Considerações Especiais: | Não se aplica. |

## Teste de Integração, Cadastro de Edição com Programação

|  |  |
| --- | --- |
| Objetivo da Técnica: | Cadastrar uma programação do evento SATI |
| Técnica: | Observados os Testes de unidade realizados em Edição e sendo eles válidos, o mesmo sendo realizado a uma programação, o teste consiste basicamente em registrar uma programação que pertence a está edição e a partir de manipulação de funções verificar que a programação está mesmo atrelada a edição que se espera. Também excluir programação existente e verificar se está alteração foi realmente mesmo persistida |
| Estratégias: | A estratégia utilizada para analisar a técnica foi o Teste de integração através do método de Caixa Branca. Para descrever o algoritmo foi utilizado o método de teste de caminho básico. |
| Ferramentas Necessárias: | * Computador com máquina virtual JAVA instalado. * Acesso à o banco de dados MySQL. * JUnit. |
| Critérios de Êxito: | A técnica suporta o teste de:   * Integração, testando o método de cadastro de uma nova edição no banco de dados, cadastro de uma programação e verificação de correspondência entre ambas. * Caixa Branca, que foi utilizado para descrever o algoritmo utilizado.   Quando dois objetos possuem dados corretos, deverá ser possível cadastrar o relacionamento entre classes no banco de dados. |
| Considerações Especiais: | Não Aplica. |

## Teste de Integração, Cadastro de Pessoa com Programação

|  |  |
| --- | --- |
| Objetivo da Técnica: | Cadastrar uma programação do evento SATI |
| Técnica: | Observados os Testes de unidade realizados em Pessoa e sendo eles válidos, o mesmo sendo realizado a uma programação, o teste consiste basicamente em registrar uma programação que tem um responsável, e a partir de manipulação de funções verificar que a programação está mesmo atrelada ao seu responsável correto, verificado essa informação a exclusão de um responsável deve ser permitida sem que a programação deixe de existir. |
| Estratégias: | A estratégia utilizada para analisar a técnica foi o Teste de integração através do método de Caixa Branca. Para descrever o algoritmo foi utilizado o método de teste de caminho básico. |
| Ferramentas Necessárias: | * Computador com máquina virtual JAVA instalado. * Acesso à o banco de dados MySQL. * JUnit |
| Critérios de Êxito: | A técnica suporta o teste de:   * Integração, testando o método de cadastro de uma nova programação no banco de dados, e verificação de correspondência entre seu responsável. * Caixa Branca, que foi utilizado para descrever o algoritmo utilizado.   Quando dois objetos possuem dados corretos, deverá ser possível cadastrar o relacionamento entre classes no banco de dados. |
| Considerações Especiais: | Não Aplica. |