Plano dE TESTE ESTRUTURAL

Sistema de Avaliação do Docente – SAD

Versão 1.0

**Goiânia, 20 de Junho de 2013**

**Revisões**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versão | Descrição | Ator |
| 20/06/2013 | 1.0 | Criação do Plano de Teste Estrutural. | Thais Cardoso |

# Introdução

O Plano de Teste Estrutural tem como objetivo o planejamento, especificação, documentação, recursos e programação das atividades de teste estrutural do sistema.

# Propósito

Conforme Plano de Teste Funcional

## Itens de Teste

Conforme Documento de requisitos

# Abordagem e Critérios

Neste capítulo serão definidas as abordagens adotadas para testar o produto de software bem como os critérios de entrada e saída.

# Abordagem de testes

Para efetuar os testes estruturais do SAD será usada a seguinte abordagem:

* Teste Caixa-Branca: Técnica de teste que avalia o comportamento interno do componente de software (Figura 4). Essa técnica trabalha diretamente sobre o código fonte do componente de software para avaliar aspectos tais como: teste de condição, teste de fluxo de dados, teste de ciclos e teste de caminhos lógicos (PRESSMAN, 2005)

Apresenta-se a seguir o tipo de teste estrutural que será utilizado neste sistema:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de teste** | **Abordagem** | **Objetivos** | **Metodologia e Ferramentas** |
| Teste de Unidade | Caixa Branca (Estrutural) | - Testa a unidade  - Garantir máxima cobertura do código-fonte  - Garantir o processamento de diferentes caminhos | - Framework de testes unitários  - Framework de grau de cobertura do código |

* 1. **Critérios**

Ao definir os critérios de teste, definem-se medidas claras para avaliar e conduzir os testes de software. Segue a tabela com os critérios definidos para esse plano de teste:

|  |  |
| --- | --- |
| **Objetivo do Teste:** | O objetivo dos testes unitário é testar a menor funcionalidade existente do software, ou seja, isolar parte dos códigos e métodos, e analisar se essas funcionalidades tenham o retorno esperado mediando a um valor informado |
| **Técnica:** | 1.       Definir uma **lista** de tarefas a implementar (o que testar).  2.       Escrever uma classe (**test case**) e implementar um método de teste para uma tarefa da lista.  3.       Rodar o **JUnit** e certificar que o teste falhou.  4.       Implementar o código mais **simples** que rode o teste.  5.       **Refatorar** o código para remover a duplicação de dados.  6.       Caso necessário, escreva mais um teste ou **refine** o existente.  7.       Monitorar a cobertura do código, para verificar se o teste está sendo coberto pelo teste unitário.  8.       Faça esses passos para toda a lista de tarefas.  http://www.devmedia.com.br/imagens/javamagazine/mpjuiiujfig02.jpg |
| **Critério de Início:** | * Especificação Funcional e de Requisitos disponíveis e aprovados * Requisito e regra de negócio que serão codificados |
| **Critério de completeza e sucesso** | Cobertura do teste:   * O teste criado para a tarefa passou. * Obteve-se 100% de cobertura do código |
| **Critério de Suspensão e Retomada** | Suspensão   * O teste unitário falhou   Retomada   * Código revisado e corrigido |
| **Critério de Finalização:** | * Todos os testes planejados foram executados. * Todos as falhas identificadas foram resolvidas * Todos os testes passaram * Todo o código foi coberto |

1. **Recursos**
   1. **Recursos Humanos**

Conforme Plano de Teste Funcional

* 1. Ambiente

Abaixo é especificado as características do ambiente necessário para execução dos testes, ou seja, as infraestrutura de hardware e software.

A infraestrutura de hardware pode ser verificada no Plano de Configuração.

|  |  |
| --- | --- |
| **Ferramenta** | **Descrição** |
| JUnit | Framework com suporte para automação de testes unitários |
| EclEmma | EclEmma é um plugin de cobertura de código Java para o Eclipse |

1. **Riscos**

Conforme Plano de Teste Funcional.