Execução de testes no ConPorta_022018

Antonio Arlis Santos da Silva, Beatriz Nogueira Carvalho da Silveira, Breno de Melo Gomes
Adailton Ferreira de Araujo[1], Juliano Lopes de Oliveira[2]
Universidade Federal de Goiás - Regional Goiânia
Curso Graduação em Bacharel Engenharia de Software
INF0089 - Integração 2[1], INF0150 - Prática em Engenharia de Software[2]

Resumo: *Neste relatório consta-se os detalhes de execução dos testes aplicado às funcionalidade do projeto CONPORTA_022018, salientamos que o escopo deste relatório abrange às funcionalidades descritas nos casos de uso: 1) IdSoft - Identifica-se no Software, 2) EdiTemTim - Editar Tempo de Time out, 3) ConAcess - Controle de Acesso. Apresentando objetivo, métodos/técnicas, ambiente e resultado.*

Palavras chaves: *Massa de dados, Testes de Funcionalidade, Estratégia de Teste.*

1. Introdução

Existem n técnicas para se testar um software, mas o objetivo das n técnicas são as mesmas: assegurar o perfeito funcionamento de alguns aspectos do software.

Segundo a norma IEEE 610.12-1990, as técnica são procedimentos técnicos e gerenciais que ajudam a avaliação e a melhoria do processo. Para o XP teste é uma parte de sua metodologia, definida em TDD (Test Driven Development) testes são criados para falha, já que ainda não temos a implementação da funcionalidade em questão e, em seguida, implementar a funcionalidade para o teste passar. Desta forma os testes são usados para clarear a ideia em relação ao que se deseja em relação ao código.

Para: RIOS, Emerson & MOREIRA, Trayahú, 2003; Teste Caixa Preta (Black Box) Visa avaliar a funcionalidade e a aderência aos requisitos, em uma ótica externa ou do usuário, sem basear em qualquer conhecimento do código e da lógica interna do componente testado, Complementa definido Testes funcionais como grupos de testes que avaliam se o que foi especificado foi implementado, normalmente servindo de base de um processo de verificação automática. Neste teste devem ser considerados os valores extremos que examinam os limites do sistema.

Partido dos princípios de boas práticas definidos acima, para a execução desse projeto buscou se aplicar as técnicas em conjunto para a construção consistente de um software.

2. Execução

Na execução do projeto após a definição e verificação dos artefatos gerados nesta etapa (Especificação de Design Funcional), foi definido os testes de funcionalidades dos casos de uso, para a definição desses testes foi definido as estratégias abaixo:

- Pontos Críticos, valores limites e valores inteligentes:
  Parte do princípio de testar os os limites inferiores e superiores, valores próximos aos limites e valores válidos e inválidos dentro de uma faixa de arredondamento (valores inteligentes, definição utilizada em algumas literaturas).
- Reparação por responsabilidade:
  Definição que parte do princípio de responsabilidades únicas e coerentes (Foi

utilizado esse conceito por uma limitação do projeto onde para exercitar o framework utilizado seria necessário a implementação de um servlet) desta forma foi possível realizar os testes nos controladores que fornecem dados para o framework e realizando mocki para as chamadas e respostas do framework, assim seria possível garantir a acurácia dos controladores e verificar o comportamento do framework.

Assim foi definido uma massa de dados otimizada para exercitar as funcionalidades, usando a técnica de teste caixa preta onde não é necessário conhecer detalhes de implementação para elaboração dos testes, adaptando o TDD para construção de funcionalidade partindo de testes de funcionalidade e não unitário como na definição literária da prática, foi possível definir um novo modelo de desenvolvimento que abstraiu a lógica da funcionalidade, remetendo o conhecido conceito de computação dividir para conquistar.

3. Resultados

No final do desenvolvimento foi possível verificar a construção de software com menos adequação de contexto como podemos verificar na imagem 1, após a execução dos testes as tabelas são criadas no banco de dados, conforme planejado no dicionário de dados.

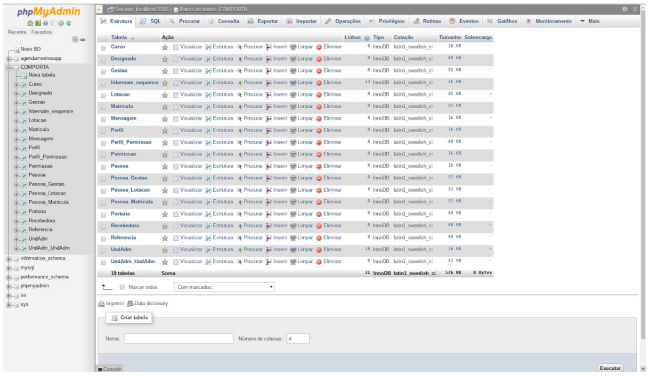

Imagem 1 - Tabelas criadas

Nas imagens 2, 3 e 4 observa-se a execução com sucesso do caso de uso EdiTemTim é os reflexos da execução no banco de dados, conforme requisito definido na descrição do caso de uso em questão, os dados apresentados na imagem 4 foram alterados e nos caso onde o fluxo de execução foi tido como sucesso e os dados não alterados nos casos onde o fluxo de execução gerava exceção em relação a imagem 3, observa-se que os dados alterados na imagem 4 são exatamente os definidos na imagem 5 que representa o caso de teste executado, os dados não alterados na imagem 4 representa a segunda seção do caso de teste que está representado na imagem 6.

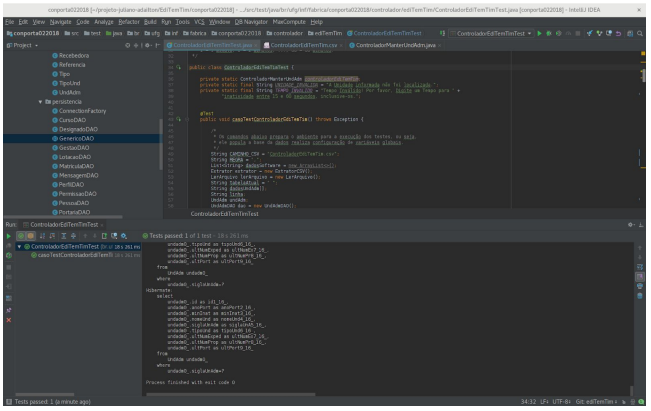

Imagem 2- Execução do CU EdiTemTim

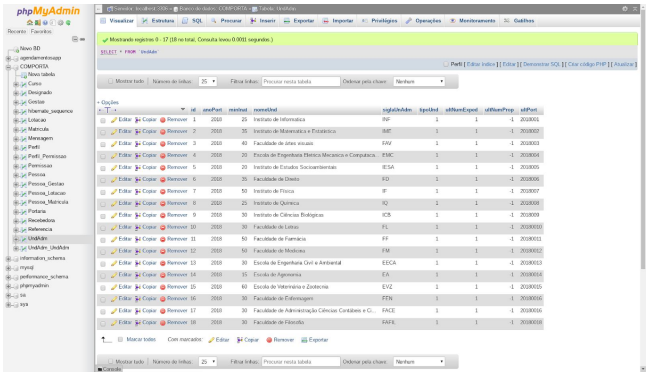

Imagem 3 - Banco de Dado original

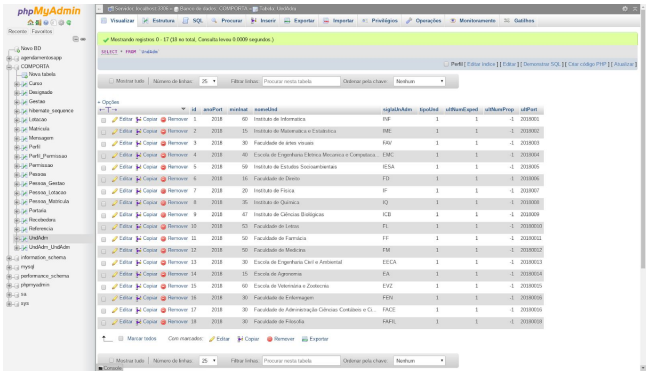

Imagem 4 - Banco de Dado Alterado

```
/*              -------->>> SEÇÃO 1 - CENÁRIOS DE SUCESSO  <<<---------                */

controladorEdiTemTim = new ControladorManterUndAdm();

/*
 * Cenário que testa os limites.
 */
controladorEdiTemTim.editarTimeOut(60, "INF");
controladorEdiTemTim.editarTimeOut(15, "IME");

/*
 * Cenário que testa os valores proximo ao limite válidos.
 */
controladorEdiTemTim.editarTimeOut(59, "IESA");
controladorEdiTemTim.editarTimeOut(16,"FD");

/*
 * Cenário  que testa valores estratégicos válidos.
 */
controladorEdiTemTim.editarTimeOut(20, "IF");
controladorEdiTemTim.editarTimeOut(35, "IQ");
controladorEdiTemTim.editarTimeOut(47, "ICB");
controladorEdiTemTim.editarTimeOut(53, "FL");
```

Imagem 5 - Cenários de sucesso

```
/*              -------->>> SEÇÃO 2 - CENÁRIOS DE EXCERÇÕES  <<<---------              */

/*
 * Cenário que testa os valores limites superiores e inferiores próximo.
 *
 * Espera se "True" no lançamento da exceção.
 */
try {
    controladorEdiTemTim.editarTimeOut(61,"FF");
} catch (Exception e) {
    Assert.assertEquals(TEMPO_INVALIDO, e.getMessage());
}
try {
    controladorEdiTemTim.editarTimeOut(14, "FM");
} catch (Exception e) {
    Assert.assertEquals(TEMPO_INVALIDO, e.getMessage());
}

/*
 * Cenário  que testa valores estratégicos inválidos.
 *
 * Espera se "True" no lançamento da exceção.
 */
try {
    controladorEdiTemTim.editarTimeOut(68, "EMC");
} catch (Exception e) {
    Assert.assertEquals(TEMPO_INVALIDO, e.getMessage());
}
try {
    controladorEdiTemTim.editarTimeOut(100, "EECA");
} catch (Exception e) {
```

Imagem 6 - Cenários de Exceção

Na imagem 7 notamos a execução dos casos de uso, ConsAcess, RefLogin e IdSof para esse caso de uso é aplicado a mesma estratégia de valores do caso de uso descrito acima com a particularidade da utilização do framework para gerenciamento de controle de acesso e por esse motivo foi dividida em duas partes sendo que a primeira executou os controladores e a segunda executou o framework.

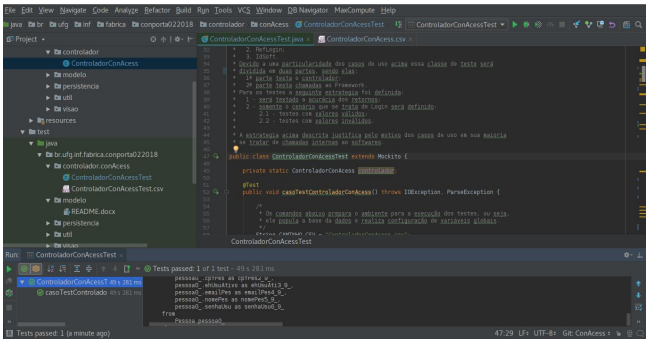

Imagem 2- Execução do CU ConAcess

4. Conclusão

Com a integração das técnicas definidas acima foi possível realizar um projeto com que pode-se garantir um nível confiável de qualidade com a utilização de pouco recurso com testes, pois foi possível definir um teste de funcionalidade automatizado, que atende ao mesmo tempo como teste de unidade e teste de integração, sendo teste de unidade, por exercitar o fluxo de execução atendendo todos os caminhos críticos do caso de uso e de integração, por exercitar a funcionalidade como um todo integrado as n classes que são percorridas para alcançar o objetivos da funcionalidade. Ficando como observação que testes pensados na otimização pode ter n funcionalidades, exercitar uma funcionalidade atendendo diversos aspectos para garantir um nível de qualidade confiável utilizando pouco recurso.

Observação: Os testes foram aplicados em ambiente controlado.