



# Execução de testes no ConPorta\_022018

Antonio Arlis Santos da Silva, Beatriz Nogueira Carvalho da Silveira, Breno de Melo Gomes Adailton Ferreira de Araujo<sup>[1]</sup>, Juliano Lopes de Oliveira<sup>[2]</sup>

Universidade Federal de Goiás - Regional Goiânia

Curso Graduação em Bacharel Engenharia de Software

INF0089 - Integração 2<sup>[1]</sup>, INF0150 - Prática em Engenharia de Software<sup>[2]</sup>

**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.

phpMyAdmin	M Estatus A SQ	L & Procurar	☐ Consults	5 Deporter	in Importor	<i>∂</i> Operações	as Prove	ligion 🚵 Retiro	a 🕙 Eventos	N Gatho	s ® Monitoramento	≠ Mais	
cente Favorios	Tabela	Ação				Links	g Tipo	Calacia	Tamanho Sobreca				
Novo BD	D Carso	#   Youize	Je Estatura (g	Procurar \$4 Inser	r 🖶 Limpar 🕻			latind_swedsh_ci	16.13				
agendamentosapp	☐ Designado	★ □ Yesolzar	Je Estados e	Procurar St Inser	e @ Limpor	Elmisar	a lanc08	bond_swedsh_ci	0.00				
CONTROL  New Model  Control  C	□ Gestas	# [] Vessizar	De Estados de	Procurar 34 Inser	r 😭 Limpor 🔞	Ekrosar	4 Inno08	kind_swedsh_ci	12 13				
	☐ hibersate_sequence	right ⊞ Visualizar	Je Domana 🝖	Procurar 24 Insec	i Milimpor (	Direitar	13 InnoCE	bin1 swedsh ci	16 83				
	□ Lotacae	* E Veseizer	Je Estatus de	Procurar 👪 Inser	r @ Limper 4	Elminar	4 Inno08	litin1_swedsh_ci	32 13	7			
	□ Matricula	r ⊡ Vesskrar	gennam e	Procurar 🚰 Insor	r (m) Limpor (	Deniar	<sup>0</sup> InnoCEI	bin1_usedsh_ci	22 13				
	□ Mensagers	rk ⊞ Veralize	Mitmus de	Procurar Si Inser	r 📾 Limper 🧯	Director	4 Innotes	brin1_swedsh_ci	16 10				
	- Perfit	right ☐ Virginitizer	ge transma le	Proper Siles	e Slimper C	Urmar	<sup>0</sup> Innotti	biin1_swedsh_ci	16 83	-			
	p Perfit Permissan	± ⊞Yeator	Michigan of	Procurar 🚰 Inser	r 🕾 Limper 🛊	Dimine	9 InnoOB	latind_swedsh_ci	40.10				
	Permissas	☆ □ Youkza	ge traveurs og	Procurar \$4 Innor	e 🖨 Limpor 😜	Dreine	9 Innotts	hind_swedsh_ci	16 10	*			
	D Pesses	± ⊞Yearlor	M Extraora re	Procurar \$4 Inser	r 🖶 Limper 🕻	Elminar	9 InnoOB	brind_swedsh_ci	16 93	7			
	Person Centro	☆ □ Youkzar	Je Estatura .g	Procurer \$4 Inser	e 🖶 Limpor 🕻	Elevisor	<sup>0</sup> InnoCB	latind_swedsh_ci	32 1/8				
	Pessea Lotacae	· III Vesetzer	(A) Estatura (II)	Procurar \$4 Inser	e @ Limper &	Elminar	9 Inno08	bond_swedsh_ci	32 88				
	Pessea Matricula	ģ □ Vessira	De Estadora de	Procurar \$4 Inser	e @ Limpor &	Elevinar	<sup>0</sup> InnoCB	latind_swedsh_ci	32 13				
	() Perteria	± ⊞Yeselzer	Je Estatura ilig	Propurar 👪 Inser	r 😭 Limper 🕻	Elmisor	9 InnoOB	ktind_swedsh_ci	41.00	7			
	☐ Recebedasa	☆ □ Vessiza	De Bossons og	Procurar 🙀 Inser	r 👾 Limpor 💰	Eliminar	4 Inno08	latini_swedsh_ci	0.0				
	□ Referencia	★ [] Yessizar	De Constant in	Procurar 34 Inser	e 😭 Limper 🕻	Ekrisar	4 InnoOB	lating swedsh pi	0.0	7			
	□ UndAdre	* EVentor	Je Esman &	Procurar Si Inser	r Millimpor (	Elenius	18 JanoOB	brin1 swedsh ci	16 10				
S. Information, schema. S. Information, schema. S. Information, schema. S. Information S. Information S. Information S. Information S. Information S. Information	□ UndAdm_UndAdm	#    Venter	persona og	Procurar 👪 Inser	r 🐠 Limpar 🧯	Direitar	4 Invoice	laint_swedsh_ci	32.13				
	19 tabelas	Sorra					11 InnoDB	btin1_wedish_ci	576 KB 0 0y	104			
	Marcar todos	Com marca	dos:										
	⊕ Imprimir ∰ Data dictio	ener .											
	G Criar tabela												
	Nome:			úmero de colunas:	4								

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.

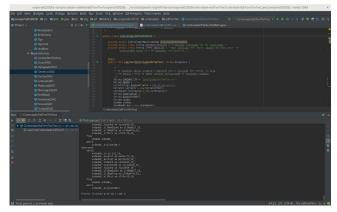


Imagem 2- Execução do CU EdiTemTim

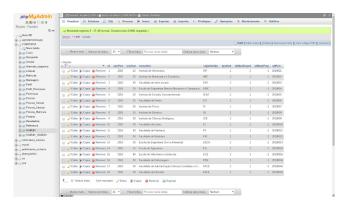


Imagem 3 - Banco de Dado original

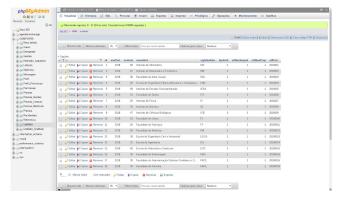


Imagem 4 - Banco de Dado Alterado





Imagem 5 - Cenários de sucesso

```
/*

* cenário que testa os valores limites superiores e inferiores próximo.

* try {
    cenário que testa os valores limites superiores e inferiores próximo.

* Espera se "True" no lançamento da exceção.

*/

try {
    centroladorediTemTim.editarTimeout(61, "FF");
} catch (Exception e) {
    Assert.assertEquals(TEMPO_INVALIDO, e.getMessage());
}
} 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(88, "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.

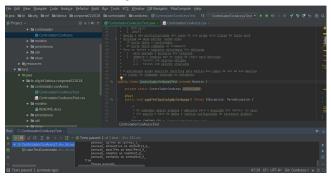


Imagem 2- Execução do CU ConAcess

#### 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 ter n funcionalidades, exercitar funcionalidade atendendo diversos aspectos para garantir um nível de qualidade confiável utilizando pouco recurso.

Observação: Os testes foram aplicados em ambiente controlado.