**Roteiro para Experimento Grupo 2**

Para este experimento é necessário realizar o download do repositório **Git:** https://github.com/AllanKaio21/experimento-testes.gitem uma pasta.

Abram o cmd (Windows) ou terminal (Linux) dentro da pasta do repositório baixado(base) e execute: *docker volume create postgres\_database2*. Verifique se a porta “880” está em uso em seu desktop, se estiver entre na pasta “base” da aplicação e dentro do arquivo “docker-compose.yml” altere no escopo “app” a porta para uma disponível. Após verificar a porta execute: *docker-compose up -d*. Acesse [http://*localhost:880*](http://localhost:8080/) *ou* [http://*localhost:<porta*](http://localhost:8080/) *alterada> no seu browser*. Com a aplicação aberta, poderão ver que são dois simples *CRUD* (*Create, Read, Update, Delete*).

Neste experimento deve se cronometrar o tempo gasto para realizar as tarefas propostas, este tempo deve ser anotado em um arquivo especifico informado.

**Iniciando o Experimento**

Antes de iniciar o experimento, deve-se **Iniciar a Cronometragem**.

Nesta parte e necessário criar os testes automáticos manualmente seguindo exemplo do teste automático já informado, sendo necessário apenas os testes funcionais, dentre os para testar o CRUD da aplicação, sendo um que aplica formulário apenas para dados do modelo, teste para alterar dados já cadastrados e um para deletar dados cadastrados de acordo com o exemplo a seguir.

A criação dos testes para os modelos, seguira o seguinte exemplo de caso de teste.

Exemplo de Caso de teste (TesteCest.php) para criação:

*Descrição: Verificar se o cadastro de informações está correto.*

*Entro na rota: teste/create*

*Preencho o campo #nome com o valor João*

*Preencho o campo #cpf com o valor 865.234.350-08*

*Preencho o campo #idade com o valor 21*

*Aplico o formulário*

*Vejo no campo #nome com o valor “João”*

*Vejo no campo #cpf com o valor “865.234.350-08”*

*Vejo no campo #idade com o valor “21”*

Exemplo de Caso de teste (TesteCest.php) para atualizar dados:

*Descrição: Verificar se atualização das informações está correto.*

*Entro na rota: teste/update*

*Preencho o campo #nome com o valor João Pedro*

*Preencho o campo #cpf com o valor 865.234.350-08*

*Preencho o campo #idade com o valor 22*

*Aplico o formulário*

*Vejo no campo #nome com o valor “João Pedro”*

*Vejo no campo #cpf com o valor “865.234.350-08”*

*Vejo no campo #idade com o valor “22”*

Exemplo de um arquivo de teste automático funcional seguindo o caso de testes acima. Sendo um teste para criação e atualização. (Todos os exemplos estão na aplicação do git na pasta: *tests/functional/testeExemple*).

<?php  
class TesteCest  
{  
 // Create  
 public function TesteCreate(FunctionalTester $I)  
 {  
 $I->expectTo('Verifica se o cadastro das informações de uma pessoa está correto.');  
 $I->amOnRoute('teste/create');  
 $I->submitForm('form',[  
 'Teste[nome]' => 'João',  
 'Teste[cpf]' => '86523435008',  
 'Teste[idade]' => '21',  
 ]);  
 $I->seeRecord('app\models\Teste', [  
 'nome' => 'João',  
 ]);  
 $I->seeRecord('app\models\Teste', [  
 'cpf' => '86523435008',  
 ]);  
 $I->seeRecord('app\models\Teste', [  
 'idade' => '21',  
 ]);  
 }  
  
 // Update  
 public function testeUpdate(FunctionalTester $I)  
 {  
 $I->expectTo("Verificar se atualização das informações está correto.");  
 $model = $I->grabRecord('app\models\Teste', array('nome' => 'João'));  
 $I->amOnRoute('teste/update',['id' => $model->id]);  
 $I->submitForm('form',[  
 'Teste[nome]' => 'João Pedro',  
 'Teste[cpf]' => $model->cpf,  
 'Teste[idade]' => '22',  
 ]);  
 $I->seeRecord('app\models\teste', [  
 'nome' => 'João Pedro',  
 ]);

$I->seeRecord('app\models\teste', [  
 'cpf' => $model->cpf,  
 ]);

$I->seeRecord('app\models\teste', [  
 'idade' => '22',  
 ]);  
 }  
}

Exemplo de Caso de teste (TesteDeleteCest.php) para deleção:

*Descrição: Verificar se a deleção de informações de uma pessoa está correta.*

*Entro na rota: teste/delete com id 1*

*Não Vejo o registro #nome com o valor “João Pedro”*

Exemplo de um arquivo de teste automático funcional seguindo o caso de testes acima. Sendo usado apenas após executar os demais testes para apagar os dados criados. Onde é feito uma pesquisa de um dado já cadastrado no banco e acessado a rota de deleção, passando o id do resultado da pesquisa.

<?php  
class TesteDeleteCest  
{  
 // Delete  
 public function testeDelete(FunctionalTester $I)  
 {  
 $I->wantTo(' Verificar se a deleção de informações de uma pessoa está correta.

');  
 $model = $I->grabRecord('app\models\Teste', array('nome' => 'João Pedro'));  
 $id = $model->id;  
 $I->amOnRoute('/Teste/delete', ['id' => $id]);  
 $I->dontSeeRecord('app\models\Teste', [  
 'id' => $id,  
 'nome' => $model->nome,  
 ]);  
 }  
}

Com base nestes exemplos, deve ser criado testes automáticos para os modelos pessoa e imóvel de forma manual, seguindo os casos de testes especificados abaixo.

Casos de testes para o modelo pessoa. Os testes de criação e atualização devem estar no mesmo arquivo (TestPessoaCest.php), como no exemplo do arquivo tests/functional/testeExemple/TesteCest.php.

Caso de teste (TestPessoaCest.php) para criação do modelo pessoa:

*Descrição: Verificar se o cadastro de informações de uma pessoa está correto.*

*Entro na rota: teste/create*

*Preencho o campo #nome com o valor João*

*Preencho o campo #cpf com o valor 865.234.350-08*

*Preencho o campo #cep com o valor* 69945000

*Preencho o campo #rua com o valor* Rua 01

*Preencho o campo #estado\_id com o valor* 1

*Preencho o campo #cidade\_id com o valor* 1

*Preencho o campo #profisao Estudante*

*Preencho o campo #complemento Perto do mercado*

*Aplico o formulário*

*Vejo no campo #nome com o valor “João”*

*Vejo no campo #cpf com o valor “865.234.350-08”*

*Vejo no campo #cep com o valor* “69945000”

*Vejo no campo #rua com o valor* “Rua 01”

*Vejo no campo #estado\_id com o valor* “1”

*Vejo no campo #cidade\_id com o valor* “1”

*Vejo no campo #profisao “Estudante”*

*Vejo no campo #complemento “Perto do mercado”*

Caso de teste (TestPessoaCest.php) para atualizar dados do modelo pessoa:

*Descrição: Verificar se a atualização de informações de uma pessoa está correta.*

*Entro na rota: teste/update*

*Preencho o campo #nome com o valor João Pedro*

*Preencho o campo #cpf com o valor 865.234.350-08*

*Preencho o campo #cep com o valor* 69945000

*Preencho o campo #rua com o valor* Rua 02

*Preencho o campo #estado\_id com o valor* 1

*Preencho o campo #cidade\_id com o valor* 1

*Preencho o campo #profisao Programador*

*Preencho o campo #complemento Perto da faculdade*

*Aplico o formulário*

*Vejo no campo #nome com o valor “João Pedro”*

*Vejo no campo #cpf com o valor “865.234.350-08”*

*Vejo no campo #cep com o valor* “69945000”

*Vejo no campo #rua com o valor* “Rua 02”

*Vejo no campo #estado\_id com o valor* “1”

*Vejo no campo #cidade\_id com o valor* “1”

*Vejo no campo #profisao “Programador”*

*Vejo no campo #complemento “Perto da faculdade”*

O teste criado para deleção deve estar em um arquivo separado (TestPessoaDeleteCest.php).

Exemplo de Caso de teste (TestPessoaDeleteCest.php) para deleção:

*Descrição: Verificar se a deleção de informações de uma pessoa está correta.*

*Entro na rota: pessoa/delete com id 1*

*Não Vejo o registro #nome com o valor “João Pedro”*

Casos de testes para o modelo imóvel. Os testes de criação e atualização devem estar no mesmo arquivo (TestImovelCest.php).

Caso de teste (TestImovelCest.php) para criação do modelo imóvel:

*Descrição: Verificar se o cadastro de informações de um imóvel está correto.*

*Entro na rota: imovel/create*

*Preencho o campo #nome com o valor Casa*

*Preencho o campo #pessoa\_id com o valor 1*

*Preencho o campo #cep com o valor* 69945000

*Preencho o campo #rua com o valor* Rua 01

*Preencho o campo #estado\_id com o valor* 1

*Preencho o campo #cidade\_id com o valor* 1

*Preencho o campo #valor com o valor 120000*

*Preencho o campo #complemento Perto do mercado*

*Aplico o formulário*

*Vejo no campo #nome com o valor “Casa”*

*Vejo no campo #pessoa\_id com o valor “1”*

*Vejo no campo #cep com o valor* “69945000”

*Vejo no campo #rua com o valor* “Rua 01”

*Vejo no campo #estado\_id com o valor* “1”

*Vejo no campo #cidade\_id com o valor* “1”

*Vejo no campo #valor com o valor “120000”*

*Vejo no campo #complemento “Perto do mercado”*

Caso de teste (TestImovelCest.php) para atualização do modelo imóvel:

*Descrição: Verificar se a atualização de informações de um imóvel está correta.*

*Entro na rota: imovel/update*

*Preencho o campo #nome com o valor Casa Grande*

*Preencho o campo #pessoa\_id com o valor 1*

*Preencho o campo #cep com o valor* 69945000

*Preencho o campo #rua com o valor* Rua 02

*Preencho o campo #estado\_id com o valor* 1

*Preencho o campo #cidade\_id com o valor* 1

*Preencho o campo #valor com o valor 140000*

*Preencho o campo #complemento Perto da faculdade*

*Aplico o formulário*

*Vejo no campo #nome com o valor “Casa Grande”*

*Vejo no campo #pessoa\_id com o valor 1*

*Vejo no campo #cep com o valor* 69945000

*Vejo no campo #rua com o valor* “Rua 02”

*Vejo no campo #estado\_id com o valor* 1

*Vejo no campo #cidade\_id com o valor* 1

*Vejo no campo #valor com o valor “140000”*

*Vejo no campo #complemento “Perto da faculdade”*

O teste criado para deleção deve estar em um arquivo separado (TestImovelDeleteCest.php).

Exemplo de Caso de teste (TestImovelDeleteCest.php) para deleção:

*Descrição: Verificar se a deleção de informações de um imóvel está correta.*

*Entro na rota: imovel/delete com id 1*

*Não Vejo o registro #nome com o valor “Casa Grande”*

Após a criação dos testes. Deve-se executa-los usando o terminal, execute: docker exec -it basic-app-1 vendor/bin/codecept run tests/functional/<Pasta>/testeCest.php --steps. Primeiro deve ser executado o arquivo (TestPessoaCest.php) em seguida o (TestImovelCest.php). Após deve se executar os testes de deleção, sendo o primeiro (TestImovelDeleteCest.php) em seguida (TestPessoaDeleteCest.php).

A execução de testes retorna resultados como *PASSED*,*FAIL* e *ERRO*. No caso de *PASSED* não foi encontrado nenhum problema para aquele tipo de teste executado, em *FAIL* ocorreu algum problema que deve ser corrigido na aplicação ou na escrita do teste, em *ERRO* o teste teve algum problema ao ser executado, que pode ser um problema de implementação do teste.

Após a execução todos os testes devem retorna PASSED, devendo verificar na aplicação se todos os dados foram deletados corretamente, assim concluindo o experimento. Em caso de erros em testes verifique os retornos de FAIL gerados na pasta *tests/\_output*. Sendo arquivos HTML contendo informações sobre o que gerou o problema.

Após encerrar a criação dos testes funcionais automáticos, deverá **Finalizar a Cronometragem** e anotar no arquivo b*asic/tests/acceptance/<seunome>/TempoGasto.txt (Exemplo: “basic/tests/acceptance/allan/TempoGasto.txt”).* o tempo utilizado para a atividade acima (Exemplo: “Tempo gasto para documentação parte 1: 10:51 minutos”).