**Roteiro para Experimento Grupo 1**

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 experimento\_giitester*. 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*).

**Iniciando o Experimento**

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

Após verificar se a aplicação está funcionando corretamente, clique em “Gii” nas opções do menu ou acesse a página ([*localhost:880*](http://localhost:8080/)*/gii)**no seu browser*. Em ***Gii Tester (Tests)*** clique no botão “Start” da ferramenta de gerar testes automáticos, na página carregada no campo de “*table name”* digite o nome da tabela “pessoa” e selecione para ser gerado o teste automático e clique em *preview*. Após carregar selecione todas as opções, e clique em *Generate*. Concluindo a geração do teste, clique em “preview”.Após verifique se na pasta de testes foram gerados os testes funcionais corretamente, dentro dos arquivos dos testes: *tests/functional/pessoa* e preencha campos não preenchidos corretamente pela ferramenta, sendo estes informados com um “TODO: <mensagem>”, realize o mesmo procedimento para a tabela “imóvel”. Para executa-los entre no Terminal e execute: *docker exec -it basic-app-1 /app/runtests*. Este comando retorna duas opções, sendo executar todos os testes e executar um especifico. *Para executar um especifico e necessário informar o caminho do teste (*exemplo de caminho: *tests/functional/pessoa/TestPessoaCest.php*)*.* Execute todos os testes e observe os resultados. Para executar os testes e necessário está na raiz do projeto.

Exemplo de teste automático funcional, aplicando um formulário a um modelo da aplicação:

<?php  
class TesteCest  
{  
 public function TesteForm(FunctionalTester $I)  
 {  
 $I->wantTo('Verify exception for form');  
 $I->amOnRoute('teste/create');  
 $I->submitForm('form',[  
 'Teste[nome]' => 'João',  
 'Teste[cpf]' => '00011122290',  
 'Teste[idade]' => '21',  
 ]);  
 $I->seeRecord('app\models\Teste', [  
 'nome' => 'João',  
 ]);  
 $I->seeRecord('app\models\Teste', [  
 'cpf' => '00011122290',  
 ]);  
 $I->seeRecord('app\models\Teste', [  
 'idade' => '21',  
 ]);  
 }

}

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, 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.

Enfim deve se executar todos os testes ao mesmo tempo utilizando o comando: *docker exec -it basic-app-1 /app/runtests*, no terminal. Obtendo como resultado PASSED em todos os testes sem exceção, e assim finalizar 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, verificar se foi gerado corretamente e ter os resultados esperados, deverá **Finalizar a Cronometragem** e anotar no arquivo b*asic/tests/acceptance/<seunome>/TempoGasto.txt* o tempo utilizado para a atividade acima (Exemplo: “Tempo gasto para documentação parte 1: 10:51 minutos”).