Descrição do Processo de Testes

Gabriel Alves Castro
Yuri Serka do Carmo Rodrigues
Vinicius Menezes Toshiyuki

19 de junho de 2019

Universidade de Brasília
yserka@hotmail.com
gabriel_alvesc1@hotmail.com
vtmsugimoto@gmail.com

1 Testes Realizados

Para este projetos foram planejados dois tipos de teste: o teste de fumaça e o teste de unidade. O teste de fumaça deve checar o produto com a visão do usuário final vendo somente as saídas do programa e sem tentar provocar erros, já o teste de unidade deve checar cada módulo, que, neste caso, são funções do programa, em seus retornos dentro do programa. Para isso uma abordagem de teste de caixa *cinza* foi escolhida: somente os retornos de cada função são testados, sem ver como funciona a função por completo, mas as partes de retorno das função são checadas para ver em quais situações os retornos poderiam causar erros.

2 Ferramenta de Testes

A linguagem usada no projeto, Go, vem com uma biblioteca padrão para testes: *testing*. A escolha da ferramenta foi feita por causa da facilidade de aprendizado por parte dos desenvolvedores, que não precisam aprender uma outra linguagem, pois é usada a própria linguagem Go, por causa da facilidade de instalação, praticamente nula, precisando somente importar a biblioteca e seguir padrões de nome, e da automação dos testes, que ajuda muito a checar se o projeto está progredindo, numa abordagem TDD, ou se as alterações aos módulos afetaram os seus comportamentos esperados sem perder tempo com a checagem manual de cada módulo e caso.

Para criar um teste o que precisa ser feito é criar uma função que comece com o prefixo *Test* seguido, normalmente, do nome da função ou funcionalidade que quer testar e que recebe como parâmetros um ponteiro para T da biblioteca *testing* (**testing.T*) em um arquivo que termine com o sufixo *_test*. As funções de teste podem estar em qualquer pacote e diretório contanto que estejam dentro da árvore do projeto.

A execução dos testes pode ser feita por meio da linha de comando *go test* que recebe como argumentos, por exemplo, quais testes devem ser realizados e onde se deve procurar pelos testes a serem executados.

Um passo a passo poderia ser:

1. Criar um arquivo *foo_test.go* no mesmo pacote da função *foo*
2. Importar a biblioteca *testing*: ***import "testing"***
3. Criar a função ***TestFoo(t *testing.T)***
4. Inserir código para obter um resultado a ser testado – como uma chamada a *Foo* e salvando seu valor
5. Comparar resultado obtido com o resultado esperado
6. No caso dos resultados esperado e obtido divergirem, apontar o erro com ***t.Errorf(erro)***, onde *erro* é uma *string* com a mensagem a ser mostrada junto ao resultado de falha do teste
7. Salvar um arquivo
8. Executar, dentro do diretório onde está o teste, ***go test 'Foo'*** (entre as aspas indicamos quais testes queremos executar, neste caso, todo teste no pacote de nome *Foo* será executado. Aspas vaziam significam que todos os testes devem ser executados)
9. Verificar a saída: os testes falhos serão apresentados com — ***FAIL TestNomeDoTeste (TempoDeExecuçãoEmSegundos)*** seguidos de qual ***Errorf*** disparou o erro, mostrando o arquivo, a linha e a mensagem que foi passada pelo programador testador como mensagem de erro

Dentro de uma mesma função de testes vários testes podem ser feitos, cada um apontado por um *Errof*.

3 Exemplo de Execução de Teste

```
$ go test 'github.com/yuriserka/Engenharia_de_Software/api/controladoras/'
--- FAIL: TestRecuperarCartoesDeCredito (0.00s)
        gestaoUsuario_test.go:40: [4] RecuperarCartoesDeCredito(01234567890) = <nil>; want map with a key-value pair
FAIL
FAIL    github.com/yuriserka/Engenharia_de_Software/api/controladoras   0.014s
$
```

Figura 1: Teste do pacote controladoras com falha

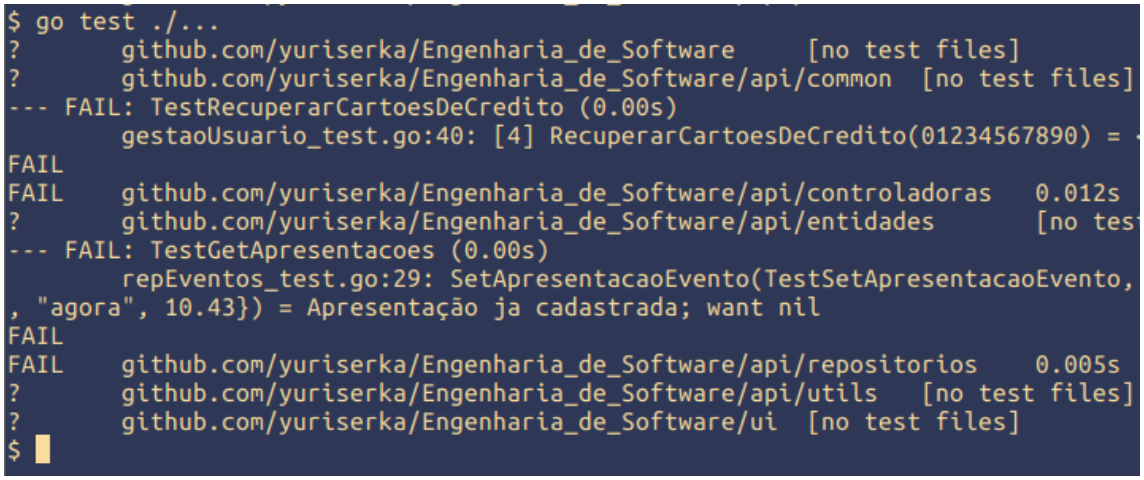

Figura 2: Com o argumento *./...* os testes são buscados no diretório atual e subdiretórios

São alguns exemplos simples de como executar os testes. Como pode ser observado, os pacotes a serem testados podem ser especificados ou não, os diretórios também. Caso nenhum argumento seja passado (*go test*), todos os testes na raiz do projeto são executados.