Universidade de Brasília – UNB

Faculdade do Gama

Engenharia de Software

VERIFICAÇÃO E VALIDAÇÃO

Walkthrough: SpaceX API - TBL 1 / Fase 3

Gama, DF

Outubro de 2019

Universidade de Brasília Faculdade UnB Gama

Autores:

Ésio G. Pereira Freitas	17/0033066	100%
Fabiana L. V. Pfeilsticker Ribas	16/0005736	100%
Fernando Ribeiro Aguilar	14/0139281	100%
Lucas D. Ferreira Nascimento	17/0050939	100%
Mikhaelle de Carvalho Bueno	15/0018673	100%
Rafael M. Gomes Ferreira	16/0142369	100%
Rogério S. dos Santos Júnior	17/0021751	100%
Youssef M. Yacoub Falaneh	17/0024334	100%

Orientador:

Dr. Ricardo Ajax Dias Kosloski

VERIFICAÇÃO E VALIDAÇÃO

Walkthrough: SpaceX API - TBL 1 / Fase 3

Gama, DF

Outubro de 2019

1. WALKTHROUGH

Walkthrough é um conjunto de procedimentos e técnicas de detecção de erros para leitura de código em grupo. A técnica baseia no princípio de que os testadores são o computador, ou seja, eles devem executar o código como se “fossem” o computador, examinando cada linha e buscando compreender àquilo que foi desenvolvido.

Em um grupo de 3 a 5 pessoas, são necessários os seguintes papéis: o moderador, que é responsável por guiar os testes, dando o direcionamento e aquilo que deve ser testado; o secretário, que fica responsável por registrar todas as anomalias ou pontos positivos do software; o testador, que é quem vai estar efetivamente testando o programa. É importante que esteja presente, também, o desenvolvedor do sistema.

O desenvolvedor não testará o programa junto com o grupo, mas será questionado sobre suas pressuposições e seu raciocínio durante o elaboração do sistema. Nesse momento são encontrados erros mais facilmente do que os casos de testes em si.

É uma técnica efetiva, pois proporciona um ambiente de competição saudável, uma vez que pessoas gostam de se destacar encontrando erros de outra pessoas, e não encontram somente o(s) erro(s), mas onde ele(s) se encontra(m) no código, o que facilita a correção. Esta técnica evidencia de 30 a 70% de erros de lógica e erros de codificação.

2. PSEUDOCÓDIGOS

2.1. spaceX.py

```
class SpaceX
begin
    run(cls)
    begin
        print menu
        read option

        if option < 1 and option > 5 then
            print "Opção inválida"
        else if option == 5 then
            exit
        else then
            showResult(option)

        print "Deseja sair da aplicação? (S/N)"
        read option

        if option == "S" then
            exit
        clearScreen()
    end

    showResult(option)
    begin
        if option == 1 then
            showAPIData("Next Launch")
        else if option == 2 then
```

```
                showAPIData("Last Launch")
            else if option == 3 then
                showAPIData("Upcoming Launches")
            else if option == 4 then
                showAPIData("Past Launches")
        end

        showAPIData(pathToConnect)
        begin
            data = ConnectTo(pathToConnect)
            print data
        end
    end
```

2.2. launch.py

```
class Launch
begin
    constructor(flight_number, mission_name, rocket, rocket_type,
        launch_success, launch_year, launch_date)
    begin
        flight_number=flight_number
        mission_name=mission_name
        rocket=rocket
        rocket_type
        launch_success=launch_success
        launch_year=launch_year
        launch_date=launch_date
    end

    printFormat()
    begin
```

```
                print getFlight_Number()
                print getMission_Flight()
                print getRocket()
                print getLaunchYear()
                print getLaunchDate()

                if launch_success is not empty then
                        print getLaunchSuccess()
        end
end
```

2.3. api_connection.py

```
class Connect
begin
        constructor(url, header=None, params=None)
        begin
                if header is None then
                        header={'Accept': 'application/json'}

                try
                begin
                        if params is not None then
                                result=requests.get(url, headers, params)
                        else then
                end             result=request.get(url, headers)
                except
                begin
                        print “Ocorreu um erro na comunicação com a API
                          SpaceX”
                end
        end
```

```
        result()
        begin
            if result.type() is Dict.type() then
                return Lauch(
                    flight_number=result.get('flight_number'),
                    mission_name=result.get('mission_name'),
                    rocket=result.get('rocket_name'),
                    rocket_type=result.get('rocket_type'),
                    launch_success=result.get('launch_success'),
                    launch_date=result.get('launch_date_utc'),
                    launch_year=result.get('launch_year')
            else
                for each instance inside result then
                begin
                    flight_number=instance.get('flight_number'),
                    mission_name=instance.get('mission_name'),
                    rocket=instance.get('rocket_name'),
                    rocket_type=instance.get('rocket_type'),
                    launch_success=instance.get('launch_success'),
                    launch_date=instance.get('launch_date_utc'),
                    launch_year=instance.get('launch_year')
                end
        end
end
```

3. INSTÂNCIAS

Para visualizar o estado do programa, o mesmo foi executado com a opção de visualizar o próximo lançamento:

spaceX.py no método: __next_launch()

Variável	Estado
option	1
connection	Connect("https://api.spacexdata.com/v3/launches/next")

api_connection.py no método: __init__()

Variável	Estado
url	https://api.spacexdata.com/v3/launches/next
headers	None
params	None
__headers	{'Accept': 'application/json'}
__response	retorno da API
__result	__response em JSON

SpaceX.py no método: __next_launch()

Variável	Estado
print(connection.result)	retorno de connection.result

api_connection.py no método: result()

Variável/Função	Estado
__result	dict
Launch(__result.get(‘flight_number’), __result.get(‘mission_name’), __result.get(‘rocket’).get(‘rocket_name’), __result.get('rocket').get('rocket_type') __result.get(‘launch_success’), __result.get(‘launch_success’), __result.get(‘launch_date_utc’), __result.get(‘launch_year’)	Instância de Launch populada com os valores de __result

launch.py no método: __init__()

Variável/Função	Estado
__flight_number	__result.get(‘flight_number’)
__mission_name	__result.get(‘mission_name’),
__rocket	__result.get(‘rocket’).get(‘rocket_name’)
__rocket_type	__result.get('rocket').get('rocket_type')
__launch_success	__result.get('launch_success')
__launch_year	__result.get('launch_year')
__launch_date	__result.get('launch_date_utc')

Como em **spaceX.py** foi chamado um o print() passando o **connection.result** e este método retorna uma instância de **Launch**, o seguinte passo será chamar o método **__str__()** de **Launch** devido ao print

launch.py no método: __str__()

Variável/Função	Estado
flight_number()	"Número do Voo: {0}".format(self.__flight_number)
mission_name()	"Missão: {0}".format(self.__mission_name)
launch_date()	"Data de Lançamento (UTC): {0}".format(date.strftime("%d/%m/%Y às %H:%M"))
launch_year()	"Ano de Lançamento: {0}".format(self.__launch_year)
rocket()	"Foguete: {0} ({1})".format(self.__rocket, self.__rocket_type)
launch_success()	"Lançamento realizado com sucesso!"

Saída desta execução:
Número do Voo: 84 **Missão:** Starlink 2 **Foguete:** Falcon 9 (FT) **Ano de Lançamento:** 2019 **Data de Lançamento (UTC):** 01/11/2019 às 00:00

4. ANÁLISE DO CÓDIGO

O código é muito bem desenhado e modularizado, é possível notar a preocupação que o programador teve ao pensar em nomes e na estruturação do código, o que torna a leitura bem simplificada. Em poucos minutos, um programador com alguma experiência em programação Web consegue entender o funcionamento com do sistema.

Contudo, foram encontrados alguns pontos que o desenvolvedor deixou passar despercebidas e que podem melhorar a experiência do usuário ao usar a aplicação:

- ➔ A falta de conexão com a internet, ainda que tenha sido tratada no construtor da classe *Connect*, não foi tratada no método *result* e quando é chamado pela classe *SpaceX* retorna um erro, pois tentar acessar o atributo *result* da instância, o qual só é preenchido quando há conexão com a *API SpaceX*. Esse problema poderia ser resolvido fazendo um teste antes de tentar acessar a variável;
- ➔ O menu de comunicação do usuário especifica que deve ser digitado para continuar a executar o sistema, mas o que o programa busca de fato é somente a letra *S* que especifica a saída do aplicação. Isso gera, ao usuário, um comportamento inesperado, o que pode gerar dúvidas ou estranheza. Dessa forma, seria necessário colocar a letra N como critério de parada e uma estrutura que ao reconhecer um caractere não esperado, pergunte-a novamente ao usuário;
- ➔ Quando é realizado o acesso a API SpaceX, o sistema fica sem resposta ao usuário do estado do programa, o que gera um sentimento de que algo está errado e leva o usuário a fechar a aplicação, a pressionar teclas aleatoriamente ou repetir a operação. Poderia ser resolvido com uma mensagem de comunicação que dissesse ao usuário o que está acontecendo;
- ➔ Quando algum caractere não esperado é digitado, o programa avisa do erro e demora aproximadamente 3 segundos para liberar o menu novamente, então caso o usuário entre uma sequência de caracteres, sendo que cada um está sendo seguido por um ENTER as mensagens de erro levarão ~3s *

N, sendo que N é a quantidade de caracteres que entraram. Poderia ser corrigido exibindo uma mensagem de digite novamente até que fosse digitado uma opção válida;

➔ Quando caracteres especiais são inseridos no programa (EOF, Interruption) o programa resulta em um erro da linguagem, o que poderia ser tratado, para que o usuário não tenha que ver esse tipo de mensagem;

5. EXECUÇÕES DO CÓDIGO

```
O que você deseja visualizar?

1) Próximo Lançamento
2) Último Lançamento
3) Próximos Lançamentos
4) Lançamentos Passados
5) Sair

Insira uma opção: 1

Número do Voo: 84
Missão: Starlink 2
Foguete: Falcon 9 (FT)
Ano de Lançamento: 2019
Data de Lançamento (UTC): 01/11/2019 às 00:00

Deseja sair da aplicação? (S/N): S
```

```
O que você deseja visualizar?

1) Próximo Lançamento
2) Último Lançamento
3) Próximos Lançamentos
4) Lançamentos Passados
5) Sair

Insira uma opção: 2

Número do Voo: 83
Missão: Amos-17
Foguete: Falcon 9 (FT)
Ano de Lançamento: 2019
Data de Lançamento (UTC): 06/08/2019 às 22:52
Lançamento realizado com sucesso!

Deseja sair da aplicação? (S/N): N
```

```
O que você deseja visualizar?

1) Próximo Lançamento
2) Último Lançamento
3) Próximos Lançamentos
4) Lançamentos Passados
5) Sair

Insira uma opção: 3

Ocorreu um erro na comunicação com a API SpaceX
Traceback (most recent call last):
  File "main.py", line 13, in <module>
    main()
  File "main.py", line 9, in main
    SpaceX.run()
  File "/home/rogeriojunior/Documents/TSW/SpaceX-API/spaceX.py", line 47, in run
    cls.__show_result(option)
  File "/home/rogeriojunior/Documents/TSW/SpaceX-API/spaceX.py", line 71, in __show_result
    cls.__upcoming_launches()
  File "/home/rogeriojunior/Documents/TSW/SpaceX-API/spaceX.py", line 117, in __upcoming_launches
    for result in connection.result:
  File "/home/rogeriojunior/Documents/TSW/SpaceX-API/connections/api_connection.py", line 40, in result
    if type(self.__result) == dict:
AttributeError: 'Connect' object has no attribute '_Connect__result'
```

```
O que você deseja visualizar?

1) Próximo Lançamento
2) Último Lançamento
3) Próximos Lançamentos
4) Lançamentos Passados
5) Sair

Insira uma opção: ^CTraceback (most recent call last):
  File "main.py", line 13, in <module>
    main()
  File "main.py", line 9, in main
    SpaceX.run()
  File "/home/rogeriojunior/Documents/TSW/SpaceX-API/spaceX.py", line 34, in run
    option = int(input("Insira uma opção: "))
KeyboardInterrupt
```

```
O que você deseja visualizar?

1) Próximo Lançamento
2) Último Lançamento
3) Próximos Lançamentos
4) Lançamentos Passados
5) Sair

Insira uma opção: Traceback (most recent call last):
  File "main.py", line 13, in <module>
    main()
  File "main.py", line 9, in main
    SpaceX.run()
  File "/home/rogeriojunior/Documents/TSW/SpaceX-API/spaceX.py", line 34, in run
    option = int(input("Insira uma opção: "))
EOFError
```

REFERÊNCIA

MYERS, Glenford J. **The Art of Software Testing**. 2. ed. rev. New Jersey: John Wiley & Sons, Inc., 2004. 255 p. ISBN 0-471-46912-2.

DEON, Victor. **SpaceX API**. 1.0. On-line, 2018. Disponível em: https://github.com/VictorDeon/SpaceX-API. Acesso em: 12 out. 2019.