**Questão 1**

Qual a diferença entre testes funcionais e não funcionais?

**R**: Nos testes funcionais validamos as funcionalidades de uma aplicação em relação aos requisitos funcionais da mesma, já nos testes não funcionais são avaliados os atributos dessa aplicação, tais como: performance, usabilidade, segurança, Integridade, etc,

**Questão 2**

Qual a diferença entre testes caixa branca e caixa preta?

**R**: Caixa Branca: Análise baseada na estrutura interna (acesso ao código) de um componente ou sistema

Caixa Preta: Análise realizada nas funcionalidades do softare com base nos requisitos.

**Questão 3**

Descreva o conceito de Smoke Test.

**R**: É um teste preliminar que visa identificar falhas simples, porém graves, que podem levar a rejeição de uma versã da aplicação.

**Questão 4**

Conceitue Análise do valor limite e Partição de equivalência.

**R**:

Na técnica de valor limite, são testados intervalos numéricos. Por exemplo um campo que aceita valores de 1 a 10, com essa técina seriam testados os limites 1 e 10 além de um valor abaixo do valor mínimo e um acima do valor máximo (0 e 11)

Na partição de equivalência os dados são divididos em grupos com dados válidos e inválidos onde o comportamento de cada grupo pode ser considerado o mesmo, então o teste é realizado com um valor de cada grupo.

**Questão 5**

Explique o conceito de pirâmide de teste.

**R**: A pirâmide de teste auxilia na determinação da quantidade de testes. Ela é dividia em 3 camadas, na sua base os testes Unitários, na camada intermediária os testes integrados e no topo os testes de Interface de usuário ou End to End.

Quanto maior o nível menos testes devem ser criados.

**Questão 6**

**A** - Qual método http devo usar em um serviço para atualizar o nome de um cliente em um cadastro?

**R**: PUT

**B** - Qual "status code" deve retornar para esta atualização?

**R**: Para atualizações realizadas com sucesso deve retonar o status code 200.

**Questão 7**

Faça uma análise do código abaixo e:

**A**- Explique resumidamente cada bloco

**B**- Faça pelo menos uma sugestão de melhoria do código

|  |
| --- |
| #bloco 1:  require 'httparty'  RSpec.configure do |config|  config.expect\_with :rspec do |expectations|  expectations.include\_chain\_clauses\_in\_custom\_matcher\_descriptions = true end |

**A**:

**B**:

|  |
| --- |
| #bloco 2:  config.mock\_with :rspec do |mocks|  mocks.verify\_partial\_doubles = true  end  config.shared\_context\_metadata\_behavior = :apply\_to\_host\_groups  config.color\_mode = true  config.before(:all) do  class HttParty  include HTTParty  end  end |

**A**:

**B**:

#bloco 3:

RSpec.describe 'Bloco 3' do

it 'Deve retornar 200 para a request valida' do

response = HttParty.get('

http://fakerestapi.azurewebsites.net/api/books')

expect(response.code).to eql(200)

end

it 'Deve retornar o content-type application/json; charset=utf-8' do

response = HttParty.get('

http://fakerestapi.azurewebsites.net/api/books')

expect(response.headers['content-type']).to eql('application/json;

charset=utf-8')

end

it 'O retorno não deve ser vazio' do

response = HttParty.get('

http://fakerestapi.azurewebsites.net/api/books')

expect(response.parsed\_response).not\_to be\_empty

end

**A**:

**B**:

|  |
| --- |
| #bloco 4: |

|  |
| --- |
| it 'Bloco 4' do  response = HttParty.get('  http://fakerestapi.azurewebsites.net/api/books')  expect(response.parsed\_response).to be\_a\_kind\_of(Array)  response.parsed\_response.each do |item|  expect(item).to be\_a\_kind\_of(Object)  expect(item["ID"]).to be\_a\_kind\_of(Integer)  expect(item["Title"]).to be\_a\_kind\_of(String)  expect(item["Description"]).to be\_a\_kind\_of(String)  expect(item["PageCount"]).to be\_a\_kind\_of(Integer)  expect(item["Excerpt"]).to be\_a\_kind\_of(String)  expect(item["PublishDate"]).to be\_a\_kind\_of(String)  end  end |

**A**:

**B**:

|  |
| --- |
| #bloco 5  it 'Bloco 5' do  response = HttParty.get('  http://fakerestapi.azurewebsites.net/api/books/1')  expect(response.parsed\_response).to be\_a\_kind\_of(Object)  expect(response["ID"]).to be\_a\_kind\_of(Integer)  expect(response["Title"]).to be\_a\_kind\_of(String)  expect(response["PageCount"]).to be\_a\_kind\_of(Integer)  expect(response["Excerpt"]).to be\_a\_kind\_of(String)  expect(response["PublishDate"]).to be\_a\_kind\_of(String)  end |

**A**:

**B**:

|  |
| --- |
| #bloco 6:  it 'Bloco 6' do  body = {  'ID': 150,  'Title': "Titulo Livro teste api",  'Description': "Descrição teste api",  'PageCount': 255,  'Excerpt': "Resumo do livro", |

|  |
| --- |
| 'PublishDate': "2019-02-06T20:35:21.207Z"  }.to\_json  response = HttParty.post('  http://fakerestapi.azurewebsites.net/api/books', body: body,  headers: {'Content-Type' => 'application/json'})  expect(response.code).to eql(200)  expect(response.parsed\_response).to eql(JSON.parse(body))  end  end |

**A**:

**B**:

**Questão 8**

Considere a seguinte criação das tabelas Cliente e Pedido:

|  |
| --- |
| CREATE TABLE Cliente (  id INTEGER PRIMARY KEY AUTOINCREMENT,  nome varchar(255),  sobrenome varchar(255)  );  CREATE TABLE Pedidos (  id INTEGER PRIMARY KEY AUTOINCREMENT,  cliente\_id int(5),  produto varchar(255),  preco decimal(5,2)  ); |

Baseado na criação das tabelas acima, responda com a declaração SQL correta

para:

**A** - Criar uma Coluna de email na tabela Cliente

**R**: ALTER TABLE Cliente ADD Email Varchar(100);

**B** - Inserir, 3 clientes na tabela Cliente com os seguintes dados: -José Silva, email: jose@cb.com.br

-João Pedro, email: joao@pf.com.br

-Pedro Silva, email : pedro@ex.com.br

**R**: INSERT INTO Cliente (nome, sobrenome, Email)

VALUES ('José', 'Silva', '[jose@cb.com.br](mailto:jose@cb.com.br)'),

('João', 'Pedro', 'joao@pf.com.br'),

('Pedro', 'Silva', 'pedro@ex.com.br')

**C** – Inserir um pedido para cada cliente com os seguintes dados: -Para o cliente 1: Produto: Geladeira Brastemp, Preço: 1800,00 -Para o cliente 2: Produto: Fogão Consul, Preço 850,90

-Para o cliente 3: Produto: Celular Iphone XR, Preço 3399,00

**R**: Insert into pedidos (cliente\_id, produto, preco)

values (1,'Geladeira Brastemp', 1800.00),

(2, 'Fogão Consul', 850.90),

(3, 'Celular Iphone XR', 3399.00)

D - Selecionar todos os pedidos de clientes com sobrenome Silva **R**: SELECT c.nome, c.sobrenome, p.\* FROM pedidos AS p INNER JOIN Cliente AS c ON  
p.cliente\_id = c.id  
WHERE c.sobrenome LIKE '%Silva%';

E - Apagar a tabela Pedidos

**R**:DROP TABLE Pedidos

**Questão 9**

Considere a história de usuário:

Como cliente das Casas Bahia

Quero adicionar um produto no carrinho

Para reservar meu produto

Escreva em BDD (Behavior Driven Development) os 2 cenários abaixo: A- Adicionar um Iphone no carrinho

**R: Cenário**: Adicionar um Iphone no carrinho

**Dado** que eu acesse a loja virtual das Casas Bahia

**E** realize uma pesquisa por iPhone

**E** clique no produto desejado para exibir mais detalhes

**Quando** eu clicar em comprar

**Então** o produto é inserido no meu carrinho

B- Adicionar um ventilador com a opção 110v no carrinho

**R: Cenário:** Adicionar um ventilador com a opção 110v no carrinho

**Dado** que eu acesse a loja virtual das Casas Bahia

**E** realize uma pesquisa por Ventilador

**E** clique no produto desejado para exibir mais detalhes

**E** selecione a voltagem 110v

**Quando** eu clicar em comprar

**Então** o produto é inserido no meu carrinho

**Questão 10**

Escolha a linguagem de programação de sua preferência para automatizar o fluxo abaixo referente seguinte página: https://bit.ly/3jOMrR9

Funcionalidade: Pesquisa com o QA

Como um recrutador

Quero colher dados da pesquisa

Para fazer análises mais detalhadas do QA

Cenário: Preencher pesquisa com dados obrigatórios válidos Dado que eu acesse a página da VV Test

E acesse o menu "Pesquisa - QA"

Quando eu preencher todos os campos obrigatórios

Então deve ser direcionado para uma página de sucesso

Instruções: Após desenvolver a automação, suba o projeto no GitHub (público). **Link:**

[**https://github.com/g1807/Gisela\_Dextra/**](https://github.com/g1807/Gisela_Dextra/)

**a**