

Candidata: Ana Evelyn de Holanda Fernandes

Link Github action projeto Maven

<https://github.com/evelynholanda/Utils>

Link Github action rodando pipeline

<https://github.com/evelynholanda/getnet/actions>

Respostas das questões:

1) O cliente adquiriu um novo modelo de máquina de cartão de crédito e ela aceita as bandeiras Visa, Master, Elo, Amex e Hiper e executa duas operações para cada bandeira débito e crédito. Quantos testes serão necessários para validação dessa máquina e qual técnica a ser utilizada?

Cenário:

Bandeira	Operação
Visa	Débito
Visa	Crédito
Master	Débito
Master	Crédito
Elo	Débito
Elo	Crédito
Amex	Débito
Amex	Crédito
Hiper	Débito
Hiper	Crédito

Tendo em vista o requisito acima podemos pensar primeiramente na técnica de testes de Software que é a **Partição de Equivalência**:

Seriam 5 bandeiras * 2 operações (débito e crédito) = 10 testes

A partir deste contexto eu faria algumas perguntas ao PO ou na reunião de refinamento:

1- Quais os limites específicos para cada bandeira? Existe um valor máximo ou mínimo?

Dependendo dos limites específicos para cada bandeira, podem ser necessários testes adicionais para verificar os limites.

Exemplo: Só pode valores acima de R\$ 0,00 e menores ou igual a R\$ 500,00

Neste caso específico do exemplo, eu aplicaria a Análise do Valor Limite que seria

_____ <= | _____ 0,01 _____ 500,00 | _____ > _____
invalido 0 válido válido inválido

sendo:

- (< = 0) -> valores inválidos

- (> 0 e ≤ 500) -> valores válidos
- ($> 500,01$) -> valores inválidos

2- Existe interações entre as bandeiras?

Se houver interações específicas entre as bandeiras e as operações (por exemplo, se uma operação de débito não for suportada por uma determinada bandeira), você precisará adicionar testes para cobrir essas combinações.

Neste caso eu teria que fazer a tabela de decisão para cobrir outros cenários

Conclusão: Teria ao menos 10 cenários de testes aplicando essas técnicas.

No entanto, ainda assim seriam necessários fazer testes exploratórios baseados em sessões (exemplo de 30,40m) para mapear outros cenários que não foram explícitos nos requisitos.

São exemplos de cenários alternativos:

- Há campos obrigatórios? validar se há campos obrigatórios nos cartões pra cada bandeira e se há campos obrigatórios nas operações
- Qual o formato de campo de data de validade do cartão?

2) A bandeira Visa disponibilizou um novo range de bin's no mercado que possui seis dígitos 232425 a 232460. Quais os testes deverão ser feitos desse range de bins para confirmar que está funcionando?

Confirmando a técnica que falei acima neste caso quando eu tenho um range de valores eu posso utilizar a técnica de análise do valor limite que ficaria mais ou menos assim. portanto Resposta seria a letra D

.....**232424**.....|.....**232425**.....|.....**232459**.....|.....**232460**..... |.....**232461**.....|

Valores inválidos => 232424 e 232461

Valores válidos => => 232425 a ≤ 232460

3) Cite uma abordagem desenvolvimento ágil e explique-a, responda em suas palavras

Bom, nos times que trabalhei eu utilizei a metodologia Scrum.

O scrum na verdade é um framework que visa organizar o trabalho (features, US, Tasks, trabalho) em sprints.

Normalmente essas sprints são de 2 semanas, com dailys diárias.

Nos times scrum que que passei eram formados por: 1 PO, um Gerente, 3 a 4 Desenvolvedores, 1 Devops e 2 QAs

Nós QAs, participamos de todas as cerimônias:

As features, US ou improvements eram escritas pelo PO, junto com o time de UI depois revisadas pelos Qas.

Nós utilizamos o DoR e DoD para saber as features que iam estar prontas para entrar no backlog e as que foram totalmente concluídas. Na escrita da US utilizamos BDD com a sintaxe Gherkin para escrever os critérios de aceite e a Técnica INVEST.

Além das escritas da US, os QAs participavam das reuniões de refinamento bem como da LA(Retrospectiva) e Review com o Cliente.

4) O que é CI/CD?

Continuous Integration e Continuous Delivery é uma abordagem que a gente usa para automatizar o processo de desenvolvimento.

No meu caso, nós utilizamos para:

- ligar e desligar os JOBS dos ambientes de testes e Integration
- Fazer o deploy no ambiente de testes da feature com a TAG nova, mais atual (Utilizava o jenkins e Gitlab CI)
- Subir os testes de regressão e de smoke (mais críticos) na pipeline para gerar maior confiança nos testes, acelerar o processo de feedback para os desenvolvedores dos testes que não estavam passando para possíveis correções promovendo uma resposta a mudança de forma rápida.

5) O que é TDD, BDD e ATDD e quando são aplicadas?

Todas são práticas de desenvolvimento que a princípio não deveriam ser utilizadas pelo QA em times ágeis.

TDD- desenvolvimento guiado a testes utilizado para testes de unidade para testar classes, métodos

ATDD_ também é desenvolvimento guiado a testes, mas um desenvolvimento guiado a testes de aceitação.No caso agora se espera testar uma funcionalidade. Essa abordagem fica ainda mais perto do entendimento do usuário

BDD- neste caso a escrita é mais colaborativa, mais estruturada. São testes baseados em comportamento.

E existe uma sintaxe estruturada para a escrita que é a utilização do Gherkin, que é o Dado, Quando e Então..

6) Cite uma heurística de testes utilizada para testes de front e de back e explique-as

Se os testes de frontend tiverem dentro dos requisitos eu posso utilizar algumas heurísticas como **CHIQUE** ou ALTER FACE.

Essas heurísticas eu aprendi de forma prática na mentoria do Julio de Lima

Exemplo:

C campos obrigatórios

H habilitar e desabilitar formulários

I interrupção ou ação

Q quebra de fluxos

U usabilidade dos menus

E estouro de campos

A **ALTER FACE** é ainda mais precisa pra testes WEB, principalmente aplicações mais modernas com uso de Angular ou React. Esta heurística é do Julio de Lima

Para testes de Backend se por exemplo eu tiver testando um APi eu posso utilizar a heurística **VADER**, essa já utilizei na prática em meu mais recente projeto na última

empresa que trabalhei pois foi eu mesma que fiz o planejamento de testes e utilizei essa heurística para fazer as validações dos response code, verbos, autorização na aplicação (token), dados válidos e os erros

7) Cite ferramentas de testes automatizados para Desktop, Web, API's e mobile

DESKTOP- Test Complete

WEB- Selenium Cypress, Robot dentre outras

API- Postman, RestAssured, Jmeter (performance)

MOBILE- Appium, Espresso, Robot

Prática de API

Desenvolva o script da automação seguindo as informações a seguir:

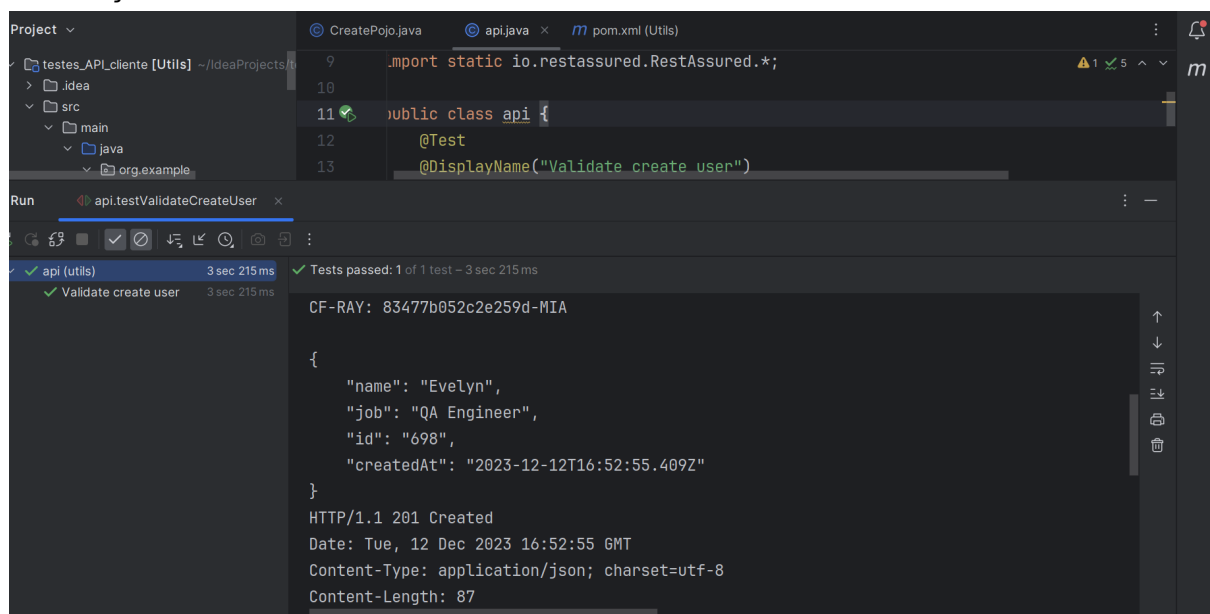
Documentação = <https://reqres.in/>

URI = <https://reqres.in/api/>

- 1) Validar o script de "CREATE" método "POST" cobertura de testes em Rest-Assured da API
- 2) Validar cobertura de Status Code, Campos obrigatórios e Contrato
- 3) Desenvolver com POJOs.

OBS1: Enviar o link do código no GitHub

VALIDAÇÃO DO CÓDIGO



The screenshot shows an IDE with a project named 'testes_API_cliente [Utils]'. The main editor displays a Java file 'api.java' with the following code:

```
9      import static io.restassured.RestAssured.*;
10
11      public class api {
12          @Test
13          @DisplayName("Validate create user")
```

The Run window shows the test 'api.testValidateCreateUser' passing successfully. The output pane displays the REST client response:

```
CF-RAY: 83477b052c2e259d-MIA
{
  "name": "Evelyn",
  "job": "QA Engineer",
  "id": "698",
  "createdAt": "2023-12-12T16:52:55.409Z"
}
HTTP/1.1 201 Created
Date: Tue, 12 Dec 2023 16:52:55 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 87
```

Response body:

```
{
  "name": "Evelyn",
  "job": "QA Engineer",
  "id": "698",
  "createdAt": "2023-12-12T16:52:55.409Z"
}
```

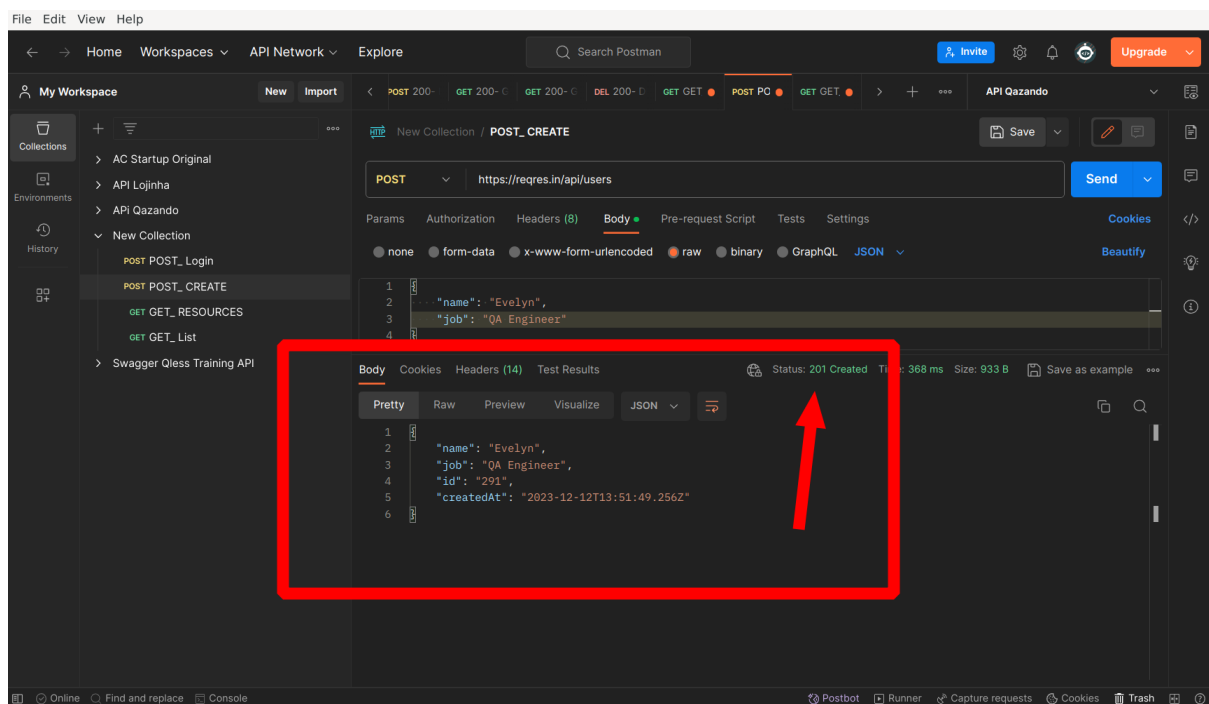
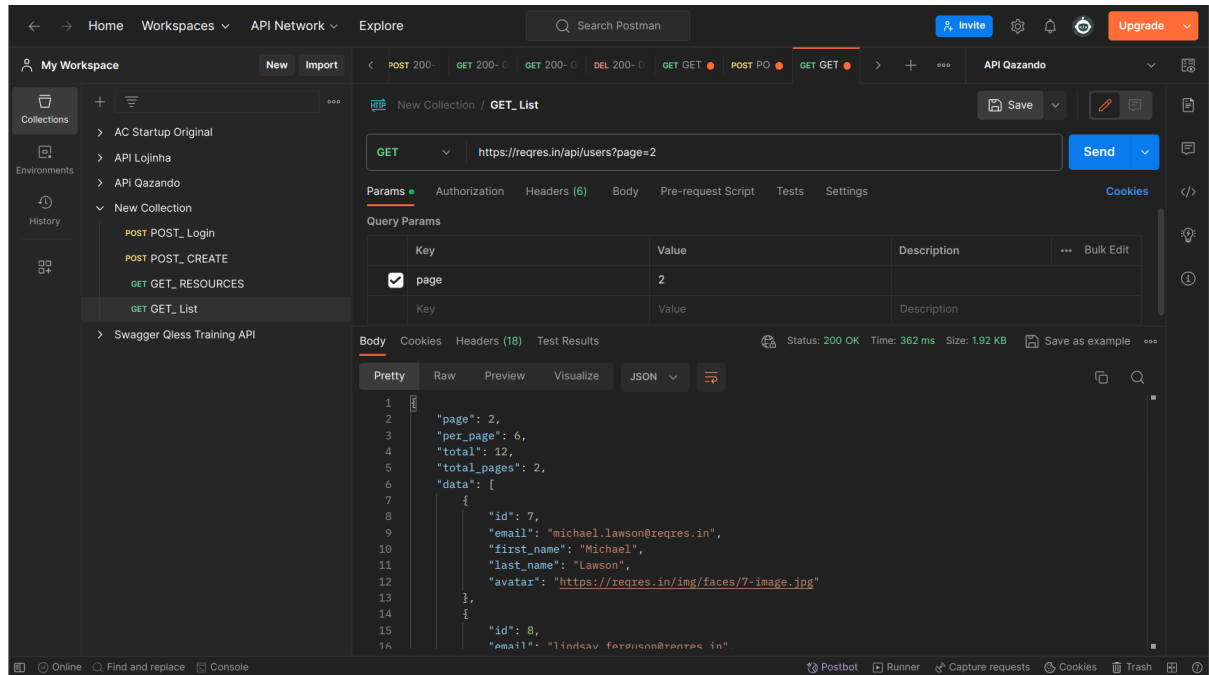
LINKS DO PROJETO DE AUTOMAÇÃO

SEGUE LINK GITHUB PARA OS TESTES EM MAVEN

<https://github.com/evelynholanda/Utils>

Para os testes inicialmente eu utilizei o Postman

Para o GET e POST



Link automação em Cypress

<https://github.com/evelynholanda/getnet>

Link Github action rodando pipeline

<https://github.com/evelynholanda/getnet/actions>