A abordagem empírica identifica o que testar com base na sua experiência.

Na abordagem sistêmica é identificado com base em técnicas e manuais criados.

**Testes com base nas regras de negócio**

**Partição de Equivalência**

Nessa técnica devemos identificar os valores de entrada e como saída teríamos a resposta se esse valor é válido/invalido, autorizado ou não.

Ex: RN1: O valor dos produtos da loja são sempre maiores que zero e nunca devem ultrapassar R $7.000,00.

Partição de equivalência é para regras de negócio, não para telas.

Variável de entrada: | Valor do Produto | .

<= R$ 0,00 R$ 0,01 até R$ 7.000,00 >= R$ 7.000,01

(inválido) (válido) (inválido)

CT01: Entrada: R$ - 5,01 | Saida: Inválido

CT02: Entrada: R$ 1000,00 | Saida: Válido

CT03: Entrada: R$ 9820,57 | Saida: Inválido

**Exercício Prático**

RN02: Um produto pode ter, no máximo, 16 componentes vinculados.

Variável de entrada: | Componentes vinculados | .

< 0 >=0 <=16 > 16

(válido) (válido) (inválido)

CT01: Entrada: -1 | Saída: válido

CT02: Entrada: 15 | Saída: Válido

CT03: Entrada: 20 | Saída: Inválido

RN03:Apenas usuários que tenham o perfil de auxiliar de estoque podem adicionar componentes a um produto.

Neste caso, o ideal é consultar quais perfis existem no sistema e se algum outro perfil é permitido fazer essa alteração e fazer até mais casos de teste..

Variável de entrada: | Perfil de usuário | .

admin auxiliar de estoque vendedor

(inválido) (válido) (inválido)

CT01: Entrada: admin | Saída: Inválido

CT02: Entrada: auxiliar de estoque | Saída: Válido

CT03: Entrada: Vendedor | Saída: Inválido

**Valores limites**

Com essa técnica, testamos todos os valores limites possíveis para ter uma porcentagem de cobertura maior.

Nesse modo o teste é realizado apenas para regras numéricas.

Ex: RN1: O valor dos produtos da loja são sempre maiores que zero e nunca devem ultrapassar R $7.000,00.

Variável de entrada: | | Valor do Produto | |

R$-0,05 | R$ 0,00 |R$ 0,01 R$ 7.000,00| R$ 7.000,01|

Invalido Valido Valido Invalido

CT01: Entrada: R$-0,05 | Saída: Invalido

CT02: Entrada: R$ 0,00 | Saida: Invalido

CT03: Entrada: R$ 0,01 | Saída: Válido

CT04: Entrada: R$ 7.000,00 | Saída: Válido

CT05: Entrada: R$ 7.000,01 | Saida: Invalido

**Exercício Prático**

RN02: Um produto pode ter, no máximo, 16 componentes vinculados.

Variável de entrada: | | Componentes Vinculados | |

| -1 0 | 1 15 | 16 17|

CT01: Entrada: -1 | Saída: Invalido

CT02: Entrada: 0 | Saída: Válido

CT03: Entrada: 1 | Saída: Válido

CT04: Entrada: 15 | Saída: Válido

CT05: Entrada: 16 | Saída: Válido

CT05: Entrada: 17 | Saida: Invalido

**Tabela de decisão**

Às vezes surgem regras condicionais baseadas em combinações de valores.

Para isso usamos a tabela de decisão

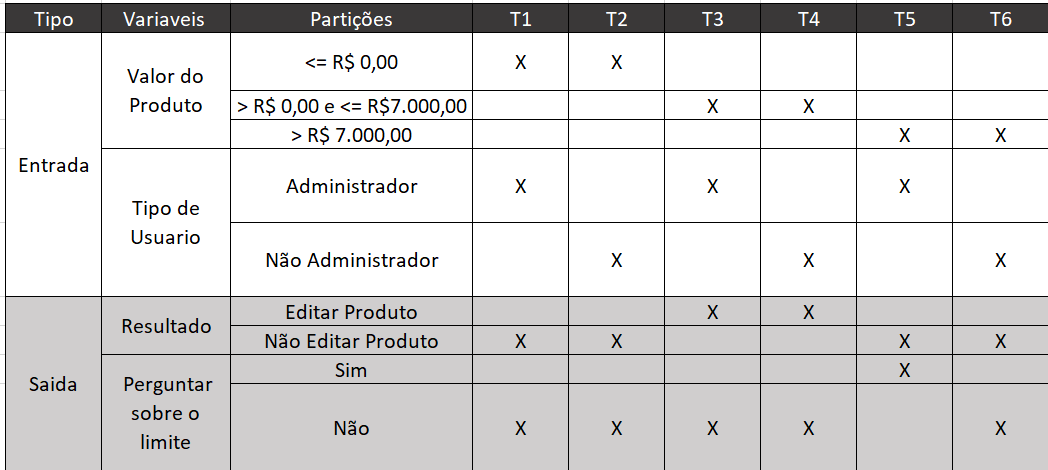
EX: RN04 Administradores verão uma mensagem perguntando se deseja alterar os limites de valor de produtos ao lidar com valores acima do limite.

| Tipo de Usuário | .

Administradores Não Administrador

As regras de negócio 01 e 04 possuem uma corelação direta, pois na RN01 tem um limite de valor estipulado em R$ 7.000,00, já na RN04 tem uma regra onde deve aparecer uma mensagem ao administrador que tentar alterar o valor para acima do permitido.

Tabela criada no excel.



**Exercício Prático**

RN02: Um produto pode ter, no máximo, 16 componentes vinculados

RN03:Apenas usuários que tenham o perfil de auxiliar de estoque podem adicionar componentes a um produto.



**Teste baseado em experiência**

ADHoc é uma forma não estruturada de seguir os testes.

Diferente dos testes baseados em regra de negócio, que são guiados através de um escopo já escrito nas RN, nos testes exploratórios usamos nossa experiência para descobrir os possíveis bugs usando nosso lado criativo e empírico, fazendo a escrita do que foi feito após realizar os testes.

**Uso de testes exploratórios com base em hipóteses**

**Como ocorrem?**

* Criação de uma hipótese.
* Planejar um ou mais cenários de teste que possam comprovar se a hipótese é verdadeira
* Aplicar os testes e observar os resultados
* Avaliar os resultados contra as hipóteses levantadas no primeiro passo.
* Repetir esse processo até que a hipótese seja comprovada (ou não)

Outra forma de testar é rever o histórico de falhas da aplicação, podemos utilizar os registros para inspirar novos testes.

**Heurísticas de teste**

Ideia de fluxo para teste exploratório:

* Campos obrigatórios;
* Habilitar / Desabilitar formulários;
* Interrupção da ação;
* Quebra de fluxos;
* Usabilidade dos menus;
* Estouro de campos.

**Uso de testes exploratórios baseados em sessão**

Composição:

1 - Sessão: Tempo predefinido entre 45 e 90 minutos, ininterruptos.

2 - Charter: Explore a inserção de produtos. Com o uso de limites inválidos. Para saber se a aplicação consegue lidar com valores não permitidos.

3 - Relatório de sessão: Um relatório que demonstra o que foi realizado durante uma sessão.

**Testes com base em riscos**

Um risco refere-se a possibilidade de algo bom ou ruim acontecer. Em testes o foco é na possibilidade das coisas ruins acontecerem. A ideia é identificar esses riscos no software e mitigá-los.

Uma forma de classificar os riscos é utilizando a **probabilidade** de algo acontecer e do **impacto** que aquilo pode causar.

Para isso usamos uma classificação de 1 a 5.

Exemplos:

**R1 P=4 I=5** Usuários não autorizados são capazes de conseguir ter acesso aos dados de um usuário autêntico.

**R2 P=5 I=1** Nossos clientes que utilizam safari e firefox não conseguem usar a nossa aplicação.

**R3 P=2 I=3** O login do site não ser acessível, impedindo 24% de usuários de usar a nossa aplicação**.**

**P = Probabilidade**

**I = Impacto**

**O Nível do Risco**

Para classificar o nível de risco é preciso multiplicar a probabilidade x impacto para saber o que devemos priorizar.

Exemplos:

**R1 P=4 I=5** nível do risco = 20

**R3 P=2 I=3** nível do risco = 6

**R2 P=5 I=1** nível do risco = 5

Conforme formos realizando os testes e verificado que o erro não aconteceu, a probabilidade pode ser classificada como um risco menor.

**Testes com base no código fonte**

Chamado de teste de unidade, geralmente feito pelo próprio dev, é um tipo de teste feito a nível de código, onde são testados os menores pedaços do projeto.

Os testes em nível de código são utilizados como contratos, com a intenção de validar um trecho do código.

Os contratos servem para que sempre que o código for alterado o teste deve ser rodado para verificar se a alteração não quebrou o código.

Cobertura de sentença é quando os testes executam todas as linhas.

Cobertura de decisão são os testes executados nas condicionais, os verdadeiros e falsos (if).