



**Teste de Software: Super Intensivo!**



# Luan Felipe Angeli

## Analista de Testes

Graduado em Informática pela UEM em 2012.  
MBA - Gestão de Projetos TI.  
Esp. Eng Soft. Enf. Teste de Software.  
Certificações CTFL e CTFL-AT.  
Experiência em Testes de software a 10+ anos.  
Analista de Testes na Matera desde 2017.



matera

# **Willgner Alexander de Souza**

## **Analista de Testes**

Graduado Análise e desenvolvimento de sistemas pela Unicesumar  
Experiência em Testes de software a 7+ anos.  
Analista de Testes na Matera desde 2016.



**matera**

# **Roberto Luis Pegoraro**

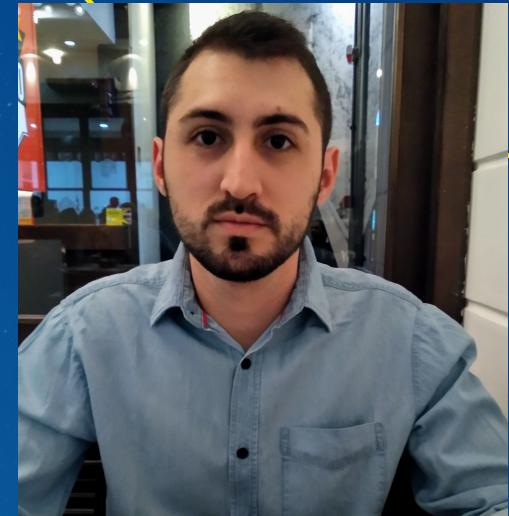
## **Analista de Testes**

Graduado em Sistemas de Informação pela UNISEP em 2011.

Esp. Desenv. de sistemas Web e Mobile.

Experiência em Testes de software a 10+ anos com foco em Automação de Testes.

Analista de Testes na Matera desde 2019.



**matera**



**Fornecemos  
Tecnologia e  
Plataformas  
robustas para o  
mercado  
financeiro**

**Faturamento de  
133  
milhões  
de reais em 2019**

**Somos  
Empresa  
**S.A**  
desde 2008**

**Somos mais de  
650  
profissionais  
distribuídos em  
São Paulo,  
Campinas, Rio de  
Janeiro, Maringá,  
Porto Alegre e  
Canadá**

# Timeline **Matera**

**1987**

**1990**

**1995**

**2000**

Início de tudo,  
como Software  
Design, sócios na  
operação

Lançamento dos  
primeiros produtos  
para o mercado  
financeiro

Pioneira no uso  
da internet

Solução para o  
SPB

**matera**

**2007**

MATERA Banco e  
MATERA Gestão  
Empresarial  
  
lançamento na  
Internet

**2008**

Entramos no  
ranking GPTW pela  
primeira

**2011**

Investimento  
de R\$8,1  
milhões junto  
ao BNDES  
  
Início do Gente  
em Ação

**2013**

Lideramos o mercado  
de Fintechs e  
alcançamos a marca  
de 300 profissionais

**2014**

Joint ventures  
Kajera e MATERA  
MVAR

**matera**

# **2016**

115 Instituições  
Financeiras e de  
Pagamentos

Integramos no  
ranking  
das 30 melhores  
empresas para  
trabalhar  
no Brasil (GPTW).

# **2017**

Patente do  
pagamento por QR  
code Offline

# **2019**

Nova sede no  
Canadá(Operações  
iniciadas)

Marca de 500  
profissionais

# **2020**

Mais de 650  
profissionais

Investimento  
Kinea

**matera**



Atendemos mais de

**120 INSTITUIÇÕES  
FINANCEIRAS.**

matera

# Roteiro

- Fundamentos de Teste de Software: Processo, modelagem, tipos, técnicas e execução de testes.
- Automação de Testes: o que e quando automatizar?
- BDD: modelando casos de testes orientados por comportamento.
- Postman: entendendo APIs e testando-as.
- Rest Assured + Cucumber: Automatizando testes de APIs/REST numa abordagem BDD.
- Selenium: Automação de testes funcionais WEB.
- Carreira de testes, desafios e certificações na área.



// Testes

## Fundamentos de Teste de Software:



# O que é testar?

- O teste do software é a investigação do software a fim de fornecer informações sobre sua qualidade em relação ao contexto em que ele deve operar. Isso inclui o processo de utilizar o produto para encontrar seus defeitos;
- “Testar é analisar um programa com a intenção de descobrir erros e defeitos” (Myers);
- Testar é exercitar ou simular a operação de um programa ou sistema;
- Testar é medir a qualidade e funcionalidade de um sistema;
- “O teste de programas pode ser usado para mostrar a presença de defeitos, mas nunca para mostrar sua ausência” (Dijkstra);



# Qual o objetivo dos testes?



- O objetivo principal em testar os softwares é de reduzir a probabilidade de ocorrências de uma falha quando o software estiver em produção, minimizando os riscos para o negócio e garantindo que as necessidades do cliente estão sendo atendidas.

# Erro/Defeito/Falha



Segundo ISTQB (2007):

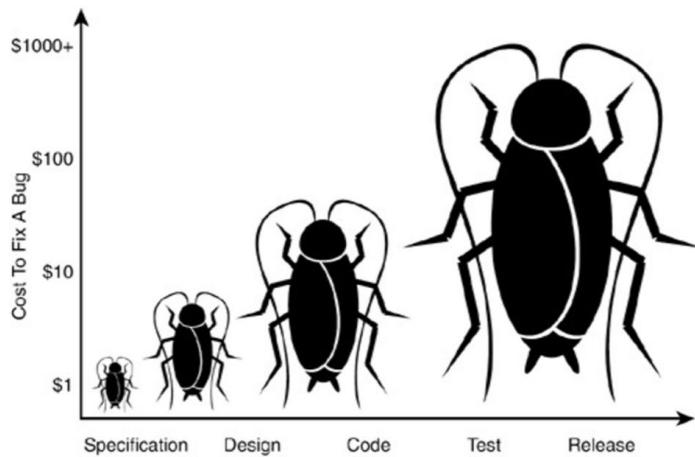
- Erro: A ação humana errada pode ser considerada um erro, que resulta em um software com defeitos. Além disso, este erro pode se tornar uma falha no sistema, fazendo com que o sistema apresente resultados incorretos.
- Defeito: Um defeito em um sistema é o resultado de um erro do programador no código fonte da aplicação. Se um determinado defeito for encontrado, quando executado pode causar uma falha no funcionamento do software.
- Falha: Uma falha ocorre quando um sistema não apresenta os resultados esperados. Deste modo uma falha é considerada uma propriedade do sistema em execução. Acontece quando um defeito gerado por um erro é executado na aplicação.

# Erro/Defeito/Falha



# Custo dos Defeitos

- Progressão de “dez, cem, mil” (Black, Rex - 1999)



# Processo de Teste

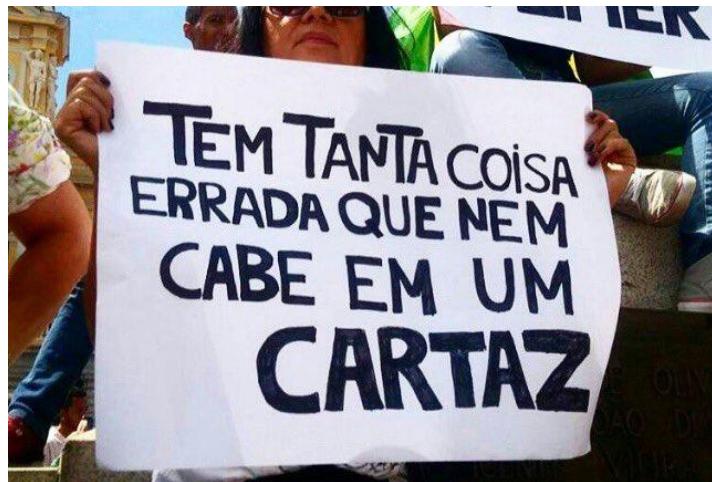
Objetivo:

- Permitir o teste de softwares usando o pessoal técnico qualificado, ferramentas adequadas e base em metodologia aderente ao processo de desenvolvimento da organização.
- As empresas precisam definir processos de testes, onde elas definem quais as atividades que serão executadas e em que momento sempre integrando essas atividades ao processo de desenvolvimento.



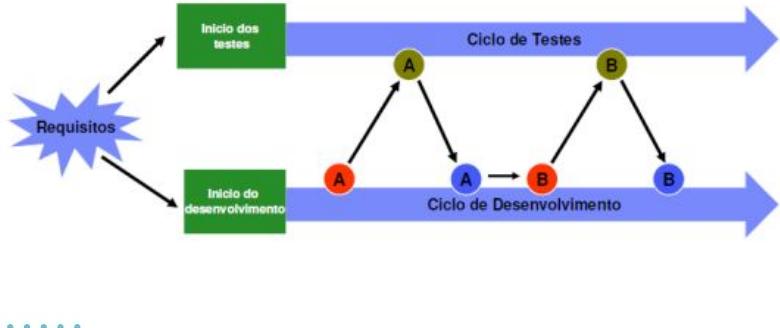
# Cenário Comum

- Testes são realizados como uma etapa do processo de desenvolvimento;
- Testes são realizados ao final do desenvolvimento;
- Prazos fazem com que o teste seja reduzido ou até mesmo eliminado;
- O objetivo é assegurar que as especificações foram construídas;
- Testes realizados por desenvolvedores ou analistas de sistemas;
- Não há garantias de que o software foi testado adequadamente;

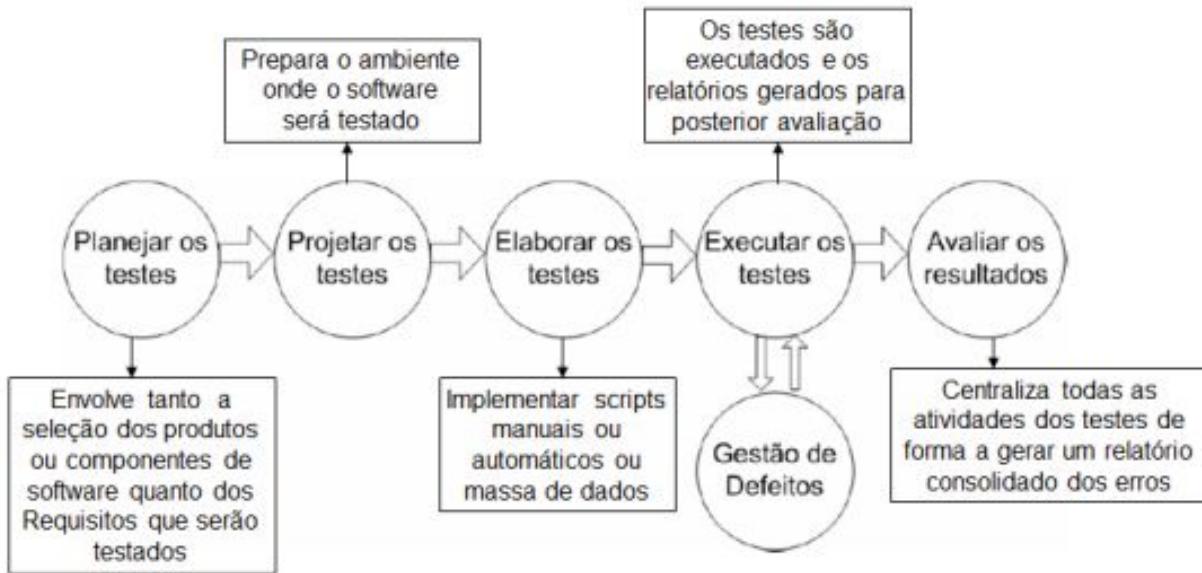


# Cenário Desejável

- O teste é tratado como um processo próprio, porém altamente integrado ao processo de desenvolvimento;
- Os testes são iniciados paralelamente ao processo de desenvolvimento;
- Testes realizados por equipes altamente qualificadas e especializadas;



# Ciclo de processo de teste



# Verificação e Validação

O processo de teste é dividido em duas grandes áreas:

## Verificação (estática):

- Analisa se o sistema atende aos requisitos funcionais e não funcionais.
- Encontra defeitos nas fases iniciais do processo de desenvolvimento.
- É realizada através de revisões e inspeções técnicas em documentos e códigos do sistema e são utilizados modelos de checklist.

## Validação (dinâmica):

- Certifica que o sistema atende as necessidades e expectativas do cliente.
- È realizado através da execução de testes como por exemplo teste funcional, teste de regressão, teste de desempenho.



Verificação

Validação

# Princípios do teste

- Princípio 1: Teste demonstra a presença de defeitos;
- Princípio 2: Teste exaustivo é impossível;
- Princípio 3: Teste antecipado (iniciar o quanto antes);
- Princípio 4: Agrupamento de defeitos (20/80);
- Princípio 5: Paradoxo do Pesticida;
- Princípio 6: Teste depende do contexto;
- Princípio 7: A ilusão da ausência de erros;



# Níveis de Teste

- Testes Unitários
- Testes de Integração
- Testes de Sistema
- Testes de Aceitação



# Testes Unitários

- Estágio mais baixo da escala de teste.
- Na grande maioria dos casos esses testes são realizados pelos desenvolvedores;
- Envolve assegurar que cada funcionalidade especificada no desenho do componente tenha sido implementada corretamente neste componente.
- O objetivo é encontrar falhas de funcionamento dentro de uma pequena parte do sistema funcionando independente do todo.
- Utiliza técnicas de Caixa Branca.



# Testes de Integração

- São executados numa combinação de componentes para verificar se juntos eles funcionam corretamente, conforme as especificações.
- Os componentes podem ser pedaços de códigos, módulos, aplicações distintas, clientes e servidores etc;
- Também costumam ser realizados pelos desenvolvedores;



# Testes de Sistema

- Executado pelos testadores para apurar se o software está fazendo exatamente aquilo que foi definido em seus requisitos funcionais e não-funcionais.
- Costuma envolver mais de um programa ou um conjunto lógico que caracterize um caso de teste.
- Têm foco no funcionamento do sistema como um todo.
- Acontece após todos os testes de integração.
- Realizado dentro de um ambiente controlado sendo o mais próximo possível do ambiente de produção.



# Testes de Aceitação

- Último nível de testes antes da implantação do software em ambiente de produção.
- Verifica se o sistema tem condições de ser implantado em produção, com base nos critérios de aceitação.
- É de responsabilidade do Cliente
- Geralmente realizado em ambiente de homologação do Cliente
- Também usado para capacitação do usuário final.



# Tipos de teste estáticos

- Revisão Informal: sem processo formal ou documentações, e a baixo custo.
- Acompanhamento: realizado em pares, varia de informal a formal e tem o propósito de aprendizado e encontrar defeitos.
- Revisão Técnica: Documentada e feita por especialista e técnicos, usa-se checklists, elabora relatórios com defeitos encontrados, recomendações, boas práticas e tem como objetivo a padronização de código e encontrar defeitos.
- Inspeção: Muito formal, com reuniões, métricas, checklists com o propósito de encontrar defeitos.



# Tipos de teste dinâmicos

- Teste Funcional: Testa os requisitos funcionais, as funções e os casos de uso. “A aplicação faz o que deveria fazer?”
- Teste de Unidade: Testa um componente isolado ou classe do sistema.
- Teste de Integração: Testa se um ou mais componentes combinados funcionam de maneira satisfatória. Há quem diga que o teste de integração é composto por vários testes de unidade.
- Teste de Segurança: Testa se o sistema e os dados são acessados de maneira segura, apenas pelo autor das ações.
- Teste de Usabilidade: Teste focado na experiência do usuário, consistência da interface, layout, acesso às funcionalidades etc.



# Tipos de teste dinâmicos

- Teste de Instalação: Testa se o software instala como planejado, em diferentes hardwares e sob diferentes condições, como pouco espaço de memória, interrupções de rede, interrupções na instalação etc.
- Teste de Integridade: Testa a resistência do software à falhas (robustez).
- Teste de Performance/Desempenho:
  - Teste de carga: Testa o software sob as condições normais de uso. Ex.: tempo de resposta, número de transações por minuto, usuários simultâneos etc.
  - Teste de stress: Testa o software sob condições extremas de uso. Grande volume de transações e usuários simultâneos. Picos excessivos de carga em curtos períodos de tempo.



# Tipos de teste dinâmicos

- Teste de Recessão: Reteste de um sistema ou componente para verificar se alguma modificação recente causou algum efeito indesejado, além de, certificar se o sistema ainda atende os requisitos.
- Teste de Manutenção: Testa se a mudança de ambiente não interferiu no funcionamento do sistema.



# Exercícios

Qual das alternativas não é um objetivo razoável do teste?

- a) Encontrar defeitos no sistema
- b) Provar que o software não tem defeitos
- c) Dar confiança no software
- d) Encontrar problemas de desempenho

Qual a proposta do teste de regressão?

- a) Verificar o sucesso das ações de correção
- b) Prevenir que uma tarefa incorreta seja completada
- c) Garantir que defeitos não foram introduzidos provenientes de alguma modificação
- d) Motivar melhores testes de unidade dos programadores

Quanto mais tarde no ciclo de vida do software um defeito é descoberto, mais caro é a sua correção. Por quê ?

- a) A documentação é fraca, então demora mais para descobrir o que o software deveria fazer
- b) Os salários vão subindo
- c) O defeito estará presente em mais documentações, códigos, testes, etc...
- d) Nenhuma das alternativas

Quando algo visível para o usuário final é um desvio em relação ao especificado, ou um comportamento não esperado, isso é chamado de:

- a) Um erro
- b) Uma anomalia
- c) Uma falha
- d) Um defeito

# O que é um caso de teste?

- É a definição de um conjunto específico de entrada de dados, condições de execução e resultados esperados, com a finalidade de avaliar os requisitos especificados do sistema.
- É um documento onde iremos informar como iremos executar, quais as informações serão utilizadas, quais as condições e quais os resultados esperados dos testes.
- Os casos de teste servem como base para que os testadores possam executar os testes.
- Um bom Caso de Teste é aquele que apresenta uma elevada probabilidade de revelar um erro ainda não descoberto.

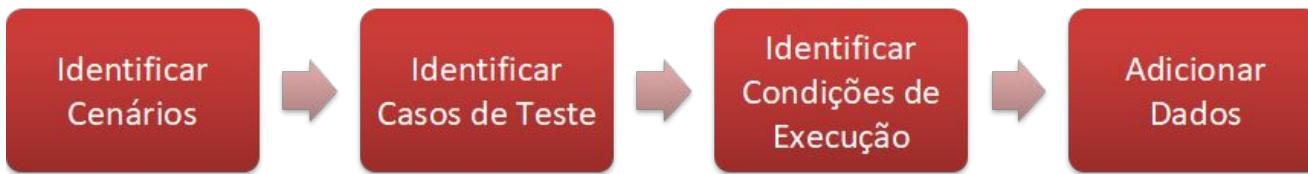


# Origem do caso de teste

- Os requisitos do sistema são originados a partir das necessidades do cliente.
- A partir dos requisitos podemos identificar os cenários de teste.
- A partir dos cenários de teste é que criamos os casos de teste.



# Elaborando casos de teste



# Exemplo de caso de teste

<b>Caso de Teste:</b>	<b>CT 001 – Senha Inválida</b>
<b>Pré-condições</b>	Estar na tela de login do Administrador
<b>Procedimento</b>	<ol style="list-style-type: none"><li>1) O ator informa uma senha inválida e preenche um login válido</li><li>2) O ator seleciona a opção OK</li><li>3) O sistema verifica se os campos obrigatórios foram preenchidos</li><li>4) O sistema verifica se o login do usuário está cadastrado no sistema e se a senha é correta;</li><li>5) O sistema exibe a mensagem "Login/Senha inválidos"</li></ol>
<b>Resultado esperado</b>	Mensagem de erro do sistema
<b>Dados de entrada</b>	Um login inválido
<b>Critérios especiais</b>	Não se aplica
<b>Ambiente</b>	Windows XP; Internet Explorer 6.0 Service Pack 2; Servidor de aplicação Apache 2.0
<b>Implementação</b>	Manual
<b>Iteração</b>	1º Iteração



# CASO DE TESTE

importância  
e como fazer



# Montando casos de teste

- Ser curioso e detalhista;
- Pensar nas vertentes e impactos;
- Objetivo do Caso de Teste é encontrar defeitos;
- Conhecer o Sistema;
- Conhecer o Negócio;
- Saber aplicar as Técnicas de Teste
- Pensar em como o usuário final usaria a funcionalidade;
- Não fugir do objeto do teste;
- Ser crítico;



# Um bom caso de teste

- Possuir um título claro, objetivo, rastreável e autoexplicativo:
- Seguir um padrão
- Ser objetivo e não exaustivo
- Tornar evidentes as situações de falha
- Ser autossuficiente
- Atingir a maior cobertura possível:
- Estar sempre atualizado
- Ser reutilizável



// Testes

(Daniele Endler) <https://medium.com/cwi-software/dicas-para-escrita-de-casos-de-teste-ccea14a7fdd9>

# Dicas

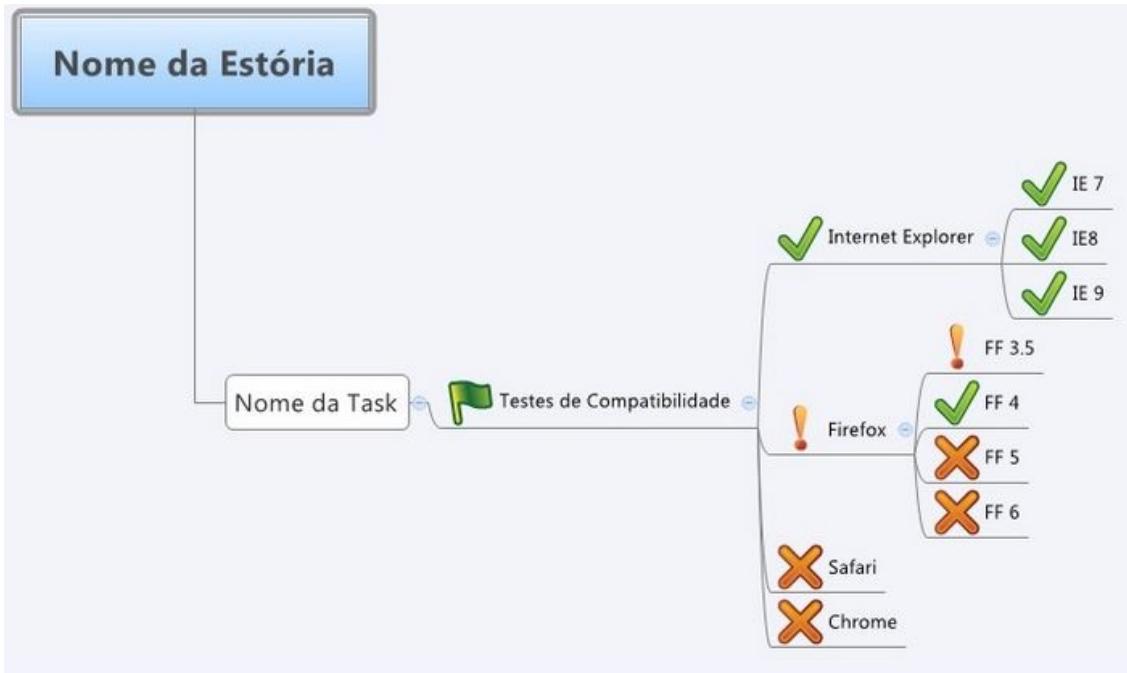
- Planejar antes de escrever
- Inserir informações que pareçam óbvias
- Sempre informar o resultado esperado
- Cuidar ações muito extensas
- Cuidar informações duplicadas e/ou contradizentes
- Cuidar a gramática
- Pode x Deve
- Revisar os Casos de Teste ao final da escrita.



// Testes

(Daniele Endler) <https://medium.com/cwi-software/dicas-para-escrita-de-casos-de-teste-ccea14a7fdd9>

# Mapas Mentais



# Exercício

Considere um sistema que calcula a autonomia de um veículo, mapeie e faça os casos de teste.

Para isso deverá haver na tela os seguintes campos:

- Quilometragem inicial: Deverá ser um EditBox numérico e inteiro, somente aceitando números positivos.  
Nele será inserido a quilometragem inicial do carro com o tanque cheio.
  - Quantidade abastecida: Deverá ser um EditBox numérico e float, somente aceitando números positivos.  
Nele será inserido a quantidade de combustível abastecida no veículo após o uso até o tanque ficar cheio novamente.
  - Quilometragem atual: Deverá ser um EditBox numérico e inteiro, somente aceitando números positivos.  
Nele será inserido a quilometragem final do carro após o uso do abastecimento.
  - Por fim um Botão que se chamará "Calcular!"  
Esse botão deverá emitir uma mensagem de falha se caso algum dos campos anteriores não estiverem preenchidos.  
Se todos os campos estiverem preenchidos e o botão for acionado, deverá ser mostrado uma mensagem de sucesso e a autonomia do veículo.  
O resultado deverá respeitar a fórmula  $(\text{Quilometragem atual} - \text{Quilometragem inicial}) / \text{Quantidade abastecida}$ )
- A autonomia deverá vir representada em (Km/l).

# Técnicas de Teste

As técnicas de testes podem ser classificadas em:

- Caixa Branca: tipo de teste no qual o código fonte é analisado. Técnicas de teste são aplicadas no código para exercitar sentenças e decisões.
- Caixa Preta: tipo de teste que considera o software como uma entidade fechada. São conduzidos na interface do software, sem preocupação com a estrutura lógica interna do software.



# Técnicas Caixa Preta

- Partição de Equivalência: Divide todas as combinações possíveis em classes.
- Exemplo: Um campo de entrada referente ao ano de aniversário aceita valores de 1900 até 2004:
  - Partição de equivalência 1: números < 1900
  - Partição de equivalência 2: números  $\geq 1900$  e  $\leq 2004$
  - Partição de equivalência 3: números  $> 2004$
  - Valores de entrada que testariam todas as partições: 1000, 2000 e 3000



# Técnicas Caixa Preta

- Valor Limite: explora os limites dos valores para preparar Casos de Teste.
  - Grande número de erros tende a ocorrer nos limites do domínio.
  - Complementa o particionamento em classes de equivalência.
- Exemplo: Um campo de entrada referente ao ano de aniversário aceita valores de 1900 até 2004:
  - Limite inferior: 1900
  - Limite superior: 2004
  - Valores de entrada que testariam os limites: 1899, 1900, 2004 e 2005



# Exercícios

Uma balança eletrônica de encomendas valida um campo numérico da seguinte maneira:

Valores Inferiores a 10kg são rejeitados, valores entre 10kg e 21kg são aceitos, valores maiores ou igual a 22kg são rejeitados

Quais das alternativas contém os valores de entrada que cobre todas as partições de equivalência?

- a) 10, 11, 21
- b) 3, 20, 21
- c) 3, 10, 22
- d) 10, 21, 22

Quais das alternativas cobrem os valores limites?

- a) 9, 10, 11, 22
- b) 9, 10, 21, 22
- c) 10, 11, 21, 22
- d) 10, 11, 20, 21

# Técnicas Caixa Branca

- Cobertura de Sentença: Avalia a porcentagem de sentenças executáveis que foram exercitadas por um conjunto de casos de testes.
- Exemplo:

```
If x < 2 then
    display_message_pouco;
    If x = 0 then
        display_message_zero;
else
    display_message_muito;
```



# Técnicas Caixa Branca

- Cobertura de Decisão: Avalia a porcentagem dos resultados de decisões provenientes das expressões condicionais IF.
- Exemplo:

```
If x < 2 then
    display_message_pouco;
If x = 0 then
    display_message_zero;
else
    display_message_muito;
```



# Técnicas baseadas em Experiência

- Baseia-se na intuição e conhecimento do testador pela sua experiência em aplicações ou tecnologias similares.
- Ampla variedade e grau de eficiência.
- Aplica-se a técnica de supor onde estão os erros (ataque a falha).



# Exercícios

```
If x = 3 then  
    display_messageX;  
    If y = 2 then  
        display_messageY;  
else  
    display_messageZ;
```

Tendo o pseudocódigo acima, quantos testes são necessários para atingir 100% de cobertura de sentença?

- a) 1
- b) 2
- c) 3
- d) 4

Tendo o pseudocódigo acima, quantos testes são necessários para atingir 100% de cobertura de decisão?

- a) 1
- b) 2
- c) 3
- d) 4

# Execução dos testes

- Depender de tudo o que foi feito anteriormente e que servirá de base para o cumprimento desta etapa.
- Uma boa estratégia de teste, uma boa análise de risco, um bom planejamento, uma boa elaboração dos casos de teste resulta em uma execução de testes bem sucedida.

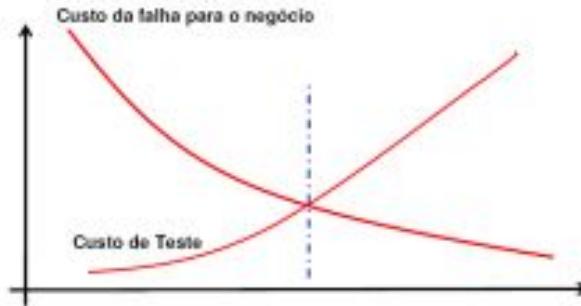
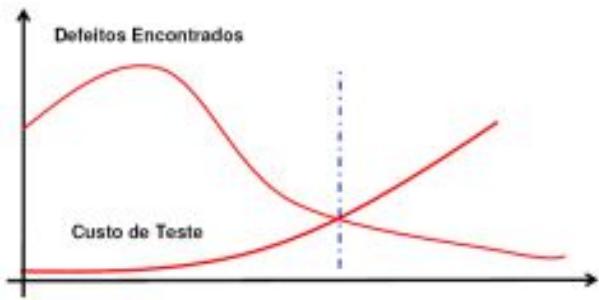


# Relatório de ocorrências

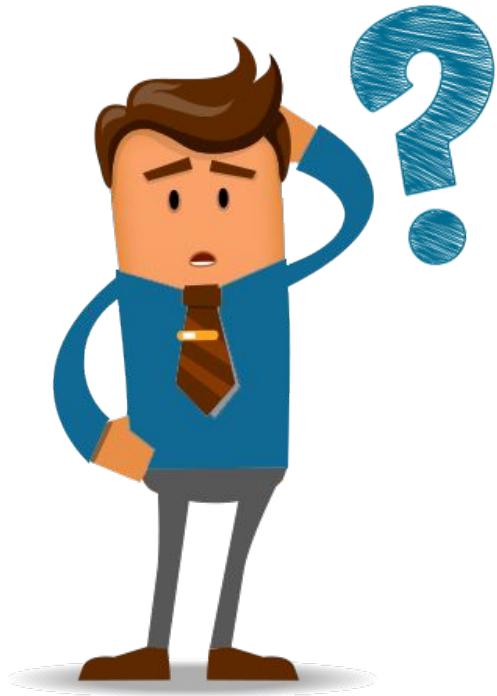
- Tem o intuito de registrar formalmente as ocorrências e seus respectivos status durante a execução dos testes.
- Deve conter informações suficientes para que quem o leia consiga entender a ocorrência e seja capaz de replicá-la.
- Ocorrências podem não ser defeitos:
  - Ocorrência: Resultado não esperado obtido durante a execução dos testes. Poderá ser mais tarde categorizada como defeito.
  - Defeitos: Resultado não esperado categorizado como defeito. Um “bug”, uma não-conformidade.



# Quando os testes terminam?



# Dúvidas?



# matera

[www.matera.com](http://www.matera.com)