Planejamento e Elaboração de um Plano de Testes

📑 Índice

👥 Squad -The Quality Ninjas

- @Gabriela Condari
- @Luis Felipe Moisyn Ferraz
- @Carolina Hoewell
- @Pedro Afonso de Alencar Silva

🎯 Objetivo do Curso

Continuação das Atividades/Cursos do Sprint 01, Semana 02.

📌 Seção 04: Planejamento de Testes (Dia 01)

- 🗓️ 31 de mar. de 2025

🤝 1. Negociação e Definição de Valor

- **Acordo flexível com o cliente:**
  - Identificar funcionalidades prioritárias vs. dispensáveis.
  - Entender o **valor para o negócio**:

        - **Vantagem competitiva** (ex.: feature exclusiva).
        - **Qualidade Mínima Viável (MVQ)**: Limite abaixo do qual o produto não é aceitável.
    - Antecipar mudanças: prazos, escopos, custos, legislações e possíveis cenários com foco em prevenção de riscos.
    - **Trade-offs**:
        - Aumentar o escopo: aumento do prazo e custo.
        - Reduzir o prazo: aumento de custo e diminuição do escopo.
        - Exemplo: Velocidade vs. segurança (aumento de custo/complexidade).

📝 2. Estrutura do Plano de Testes

- **Contrato entre equipes e cliente**:

Componente	Detalhes
Escopo	O que será ou não testado (ex.: módulo de pagamento, mas não relatórios).
Cronograma	Prazos e fases de testes (ex.: testes de regressão na semana 3).
Orçamento	Recursos (humanos, ferramentas, terceirizados).
Indicadores	Metas de qualidade (ex.: 95% cobertura de testes).
Riscos	Lista de riscos prioritários (ex.: integração com API externa) e lista de riscos do próprio projeto e do produto.
Comunicação	Como, quando e para quem reportar (ex.: status “report” semanal para clientes, registro de dailies e mudanças de escopo)

- **Padrões de documentação**:
    - **ISO/IEC 29119-3**: Modelo para planos de teste formais (útil em setores regulados).
    - **Mapa mental**: Alternativa ágil para projetos menos burocráticos.

📌 3. Requisitos e Priorização

- **Requisitos funcionais** (o *que* o sistema faz):
    - Ex.: "O usuário deve poder filtrar produtos por preço."
- **Requisitos não funcionais** (*como* o sistema faz):
    - Ex.: "O filtro deve responder em <2 segundos com 1.000 usuários simultâneos."
- **Técnicas de priorização**:
    - **Análise de riscos**: `Risco = Probabilidade x Impacto` (ex.: matriz de riscos).
        - Documente decisões passadas em um **histórico de riscos** para melhorar futuros projetos.
    - **Pareto (80/20)**: Focar em 20% dos casos que cobrem 80% dos cenários críticos.
        - Use Pareto com a matriz de riscos para priorizar onde investir esforços.
    - **Valor de Negócio (Mínima Qualidade Viável - MVQ):** O mínimo que torna o produto utilizável e competitivo.
        - A maioria dos softwares no mercado está **abaixo do MVQ** – se você atingir, já se destaca!
    - **Planning Poker**: Consenso sobre complexidade/esforço (usar cartas: 1, 2, 3, 5, 8, etc.).
        - Se **votarem ∞**, a história precisa ser quebrada em partes menores.
    - **Técnicas Combinadas (Cauda Longa + Pairwise):** Para **cenários complexos** com muitas combinações (ex.: configurações de usuário).
        - Use para testar o ‘restante’ (cauda longa) após **priorizar com Pareto**.

    - Ferramentas: PICT (Microsoft): Gera combinações otimizadas para teste / AllPairs: Reduz o número de casos mantendo cobertura.

🔄 4. Abordagens Ágeis

- **Estórias de usuário (INVEST)**:
  - Independentes, Negociáveis, Valiosas, Estimáveis, Pequenas, Testáveis.
- **Sprints e backlogs**:
  - **Product Backlog**: Todas as estórias priorizadas pelo PO.
  - **Sprint Backlog**: Itens selecionados para a iteração atual.
- **Kanban**: Fluxo contínuo com limites de WIP (ex.: máx. 3 tarefas "Em Teste").

🛠️ 5. Ferramentas e Técnicas

- **Testes combinatórios** (para "cauda longa"):
  - **Pairwise**: Testar combinações reduzidas de inputs (ex.: PICT, AllPairs).
- **Automação**:
  - Regressão, performance (ex.: Selenium, JMeter).
- **Monitoramento**:
  - Dashboards com Grafana/Kibana.

🃏 Planning Pocker

1. Dinâmica de grupo para facilitar a comunicação.
2. Facilita as discussões e votações sobre: escopo, prazos, complexibilidade e riscos.

📈 Histórico de Riscos

É importante documentar os riscos para auxiliar na tomada de decisões no futuro.

- **Importância do Histórico de Riscos**:
  - A criação e manutenção de um histórico de riscos, com matrizes de risco de cada projeto, desde o início até o fim, é crucial.
  - Esse histórico permite analisar como os riscos evoluíram, se tornaram problemas, como foram tratados e resolvidos.
- **Benefícios do Histórico**:
  - Ajuda na tomada de decisões informadas, com base em experiências anteriores.
  - Promove o aprendizado com erros e acertos, evitando repetições de falhas.
  - Facilita a estimativa de prazos, custos e recursos necessários.
  - Auxilia na identificação de riscos que poderiam passar despercebidos.
  - Compartilha experiências e aumenta a maturidade dos times da empresa.
- **Cultura de Planejamento e Matriz de Riscos**:
  - É fundamental implementar uma cultura de planejamento, com a criação de matrizes de risco no início dos projetos.
  - A retrospectiva final do projeto é uma oportunidade para avaliar a eficácia das técnicas e ferramentas de gestão de riscos.
- **Decisões e Riscos**:
  - Escolhas como optar pelo caminho mais rápido ou mais barato envolvem riscos que precisam ser compreendidos e gerenciados.
  - É essencial aprender com os erros dessas escolhas para evitar que se repitam.
- **Experiência Compartilhada**:
  - Preservar o histórico de riscos auxilia na criação de uma memória de estimativas e decisões.

- A soma das experiências dos times permite uma visão compartilhada, que ajuda a evitar armadilhas.

📊 Análise de Pareto

Priorizar os testes considerando os prazos, orçamento, pessoas e outros recursos.

1. Priorizar por impulso: de menor a maior.
2. Priorizar por atração: resolver item semelhantes.
3. Priorização da calda longa: determinar uma amostragem.

✅ Resumo Prático para Ação

1. **Defina o MVQ** com o cliente.
2. **Priorize** usando matriz de riscos e/ou Pareto.
3. **Documente** o plano (formal e/ou mapa mental).
4. **Comunique** status e bugs de forma transparente, definindo o melhor formato com a equipe.

Exemplo de Matriz de Risco:

Nome do Risco	Probabilidade. (P) 1-3	Impacto (I) 1-3	Prioridade (R=P×I)	Estratégia	Como?	Responsável	Reavaliar
Integração com API de pagamento	3	3	**9**	**Mitigar**	Testar em ambiente staging antes do deploy.	João (DEV)	Semanalmente
Perda de dados no backup	2	3	**6**	**Evitar**	Implementar verificação automática de backups.	Ana (Ops)	Mensalmente
Demora na resposta do suporte	1	2	**2**	**Aceitar**	Documentar limitação no manual do usuário.	Carlos (QA)	Por evento

🔎 Insights e Observações Gerais

- **Qualidade > Cobertura total**: Foque nos riscos críticos primeiro.
- **Comunicação clara**: Evite termos técnicos com stakeholders não técnicos.
- **Histórico de riscos**: Documente decisões passadas para melhorar futuros projetos.

📌 Seção 05: Análise, Modelagem e Implementação (Dia 02)

- 🗓️ 1 de abr. de 2025

🔍 1. Análise de Testes (O "Porquê" e "O Quê")

- **Defina cenários críticos** baseados em:
  - **Eventos sazonais**: Black Friday, picos de acesso.
  - **Ambientes**: Diferenças entre Dev, Teste e Produção (ex.: DLL Hell).
  - **Dados**: Massas de teste com resultados esperados (ex.: usuários VIP vs. comuns).
- **Técnicas de priorização**:

- **Pareto**: 20% dos casos cobrem 80% dos riscos (ex.: fluxo de pagamento).
- **Matriz de Risco**: `R = Probabilidade x Impacto` (ex.: integrações externas = alta prioridade).

✏️ 2. Modelagem (O "Como")

Escolha o tipo conforme o risco e recursos:

Tipo	Melhor Para	Exemplo
Lógica	Times experientes	"CT01 - Validar login com credenciais válidas".
Concreta	Times iniciantes	"Acessar www.learning.compass.uol → Login: user@compasso.com.br, Senha: 1234".
Data-Driven	Testes com múltiplos dados	Tabelas no Gherkin (Given/When/Then).
Keyword-Driven	Automação colaborativa	Keywords como "PreencherFormulário".

> ***Quanto maior o risco, mais detalhada a modelagem (Concreta/Data-Driven).***

⚙️ 3. Implementação (Os "Detalhes")

- **Passos claros**:

```
1. Acessar/checkout
2. Inserir CVV 123
3. Clicar em "Finalizar"
Resultado esperado: Compra aprovada.
```

- **Pré-requisitos**:
  - 🖥️ Ambiente Docker configurado.
  - 🛒 Carrinho com 1 item de teste.

🔄 4. Abordagens Ágeis (TDD/BDD/ATDD)

Abordagem	Foco	Quando Usar
TDD	Testes técnicos (devs)	Lógica complexa (ex.: cálculos).
BDD	Comportamento visível	Colaboração com PO (ex.: Gherkin).
ATDD	Critérios de aceitação	Validação entre time multidisciplinar.

- Combine **BDD + Data-Driven** para testes de UI alinhados ao negócio.

📌 5. Estrutura do Plano (Exemplo)

```
1. **Escopo**:
   - ✅ Testar: Checkout, Login.
   - ❌ Não testar: Página "Sobre Nós".

2. **Cronograma**:
   - Regressão: Toda sexta-feira (JMeter).

3. **Riscos**:
   | Risco                  | Ação                          |
   |------------------------|-------------------------------|
   | API de pagamento falha | Mockar resposta em staging.   |

4. **Suites de Teste**:
   - `Suite_Smoke`: 5 casos críticos (15min).
   - `Suite_Load`: 1.000 usuários (JMeter).
```

🔎 Insights e Observações Gerais

- **Análise de Testes ≠ Planejamento:** Planejamento define o 'porquê' (estratégia), análise define o 'como' (táticas).
- **Cenários > Casos de Teste:** Um cenário (ex.: Black Friday) contém múltiplos casos de teste (ex.: carga, cupons).
- **Modelagem Flexível = Eficiência:** Modelagem lógica (1 caso) para riscos baixos; modelagem concreta (vários casos) para riscos altos.
- **Massa de Teste ≠ Massa de Dados:** Massa de teste inclui resultados esperados; massa de dados só tem inputs.
- **BDD (Gherkin) é Universal:** Given/When/Then serve tanto para manuais quanto automação, e todos entendem.
- **TDD/BDD/ATDD São Complementares:** Use TDD para lógica complexa, BDD para regras de negócio, ATDD para validação em time.
- **Docker Resolve 90% dos 'Na Minha Máquina Funciona':** Ambientes containerizados (ex.: Docker) padronizam testes e reduzem desculpas.
- **Teste Exploratório Encontra o que Scripts Não Encontram:** Deixe 20% do tempo para testes não planejados – a criatividade descobre bugs inesperados.