Teste de Software e Garantia da Qualidade (QA)

📑 Índice

- 📑 Índice
- 👥 Squad -The Quality Ninjas
- 🎯 Objetivo do Curso
- 📌 Seção 01: Apresentação do Curso (Dia 02)
  - 🎯 Aprendizado Contínuo
  - 💼 Mercado de Trabalho
  - 🔍 Qualidade é Responsabilidade de Todos
  - 🛠️ Função do QA
  - 🌟 Soft Skills (Habilidades Comportamentais)
  - 💻 Hard Skills (Habilidades Técnicas)
  - ⚠️ Débito Técnico
- 📌 Seção 02: Introdução ao Teste de Software (Dia 03)
  - 🕰️ Uma Breve História do Teste
  - 💣 A Importância do Teste x Danos dos Bugs
  - 📜 Os 7 Fundamentos do Teste (ISTQB)
  - 🔄 Diferença entre Teste e QA
  - 🔍 Erro, Ocorrência, Defeito e Falha
  - 📊 Tipos de Testes Baseados na IEC/ISO 25010 (SQuaRE - AFUCCEMPS)
  - 🤖 Testes Manuais X Automatizados
  - 🎯 Quando Usar Cada Abordagem?
  - 🔄 Testes Tradicionais X Ágeis
- 📌 Seção 03: Atitudes de um Profissional da Qualidade (Dia 04)
    - ⚡ Pressão Organizacional
    - 🔄 Comprometido vs. Envolvido
    - ⏳ Autogerenciamento
    - 📊 Técnica GUT para Priorização em QA
    - 💬 Comunicação Eficaz
    - 🤝 Negociação (Ganha-Ganha) – Para QAs
    - 🚀 Produtividade – Métodos para QAs
    - 🔄 Fluxo Contínuo (Kanban para QAs)
    - ⏳ Técnica Pomodoro
  - 🔎 Insights e Observações Gerais

👥 Squad -The Quality Ninjas

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

🎯 Objetivo do Curso

Este documento sintetiza o conteúdo essencial do curso *Início Rápido em Teste e QA*, oferecendo uma visão abrangente e prática sobre os fundamentos da garantia de qualidade de software.

O curso foi estruturado para abordar não apenas as técnicas de teste (manuais e automatizadas), mas também os princípios estratégicos da qualidade, como a aplicação de padrões internacionais (ISTQB, ISO 25010), gestão de riscos e priorização eficiente (método GUT). Além disso, enfatiza habilidades críticas para o dia a dia do QA, incluindo comunicação assertiva, negociação em cenários de pressão e autogestão produtiva (com técnicas como Pomodoro e Kanban).

Este material busca desenvolver uma mentalidade de qualidade contínua, onde o profissional aprende a equilibrar prazos, requisitos e excelência técnica. O resultado é uma formação que prepara para os desafios atuais do mercado, capacitando os participantes a entregarem sistemas confiáveis e alinhados às necessidades dos usuários e negócios.

Este registro de conhecimento foi construído a partir das anotações individuais de todos os membros do Squad durante as aulas, onde cada participante documentou sua compreensão pessoal do conteúdo. Essas contribuições foram então cuidadosamente unificadas através de um processo colaborativo que, inegavelmente, contou com o apoio de inteligência artificial. A IA atuou como ferramenta complementar na harmonização das diferentes perspectivas, oferecendo sugestões para a estruturação do documento, aprimorando a formatação, realizando correções textuais e auxiliando na consolidação coerente das diversas anotações. O resultado final mantém a essência do aprendizado humano enquanto beneficia-se da precisão tecnológica para apresentar um material coeso e profissional.

📌 Seção 01: Apresentação do Curso (Dia 02)

- 🗓️ 25 de mar. de 2025

🎯 Aprendizado Contínuo

- **Fontes de conhecimento:**
  - **Passivas:** YouTube, redes sociais, livros/e-books, blogs.
  - **Ativas:** Cursos (pagos/gratuitos), eventos, grupos de QA, hackathons.
- **Networking:** Participe de comunidades (ex.: LinkedIn) e busque mentoria.

💼 Mercado de Trabalho

- Vagas em **Teste de Software e QA** estão em alta devido à demanda por qualidade.
- **Como se destacar:**
  - Mantenha um **portfólio** com projetos e certificações.
  - Participe de eventos e práticas colaborativas (ex.: pair testing).

🔍 Qualidade é Responsabilidade de Todos

- Envolve **POs, Scrum Masters, desenvolvedores, clientes e usuários**.
- **Práticas-chave:**
  - **Pair Testing:** QA + dev testando juntos.
  - **Critérios de aceitação claros** (definidos em equipe).

🛠️ Função do QA

- Vai além de testar:
  - **Prevenir:** Identificar riscos antes que se tornem falhas.
  - **Avaliar:** Classificar e priorizar o que é mais crítico.
  - **Mitigar:** Planejar ações para lidar com os maiores riscos identificados.
  - **Documentar lições aprendidas:** para projetos futuros.

🌟 Soft Skills (Habilidades Comportamentais)

Habilidade	Descrição
Persistência	Investigar até encontrar a causa raiz de um problema.
Curiosidade	Questionar: “Como isso poderia falhar?” ou “Há uma forma melhor?”.
Empatia	Entender as necessidades do usuário/cliente.
Pensamento Crítico	Não só apontar erros, mas sugerir melhorias.
Trabalho em Equipe	Colaborar ativamente e comunicar-se claramente.
Perfeccionismo	Focar em melhorias e em entregar um software funcional e de qualidade.
Detalhismo	Observar pequenas inconsistências ou oportunidades de melhoria.
Resiliência	Lidar com mudanças frequentes em projetos.
Foco em Solução	Identificar o que realmente importa no momento.
Organização	Lidar com múltiplas tarefas, dados e prazos.
Priorização	Definir o que deve ser tratado primeiro.
Comprometimento	Desejar o sucesso do projeto na totalidade.
Empatia	Colocar-se no lugar do cliente e compreender seu uso do sistema.
Colaboração Eficaz	Ajudar e receber ajuda / Ajudar e receber ajuda / Não tomar decisões isoladas pelo grupo
Reuniões	Participe de reuniões de progresso, dúvidas, decisões, planejamento, revisão e retrospectiva.

> ***Se não for para aplicar o que foi discutido na retrospectiva, no planejamento, melhor nem fazer.***

💻 Hard Skills (Habilidades Técnicas)

Categoria	Habilidades/Conhecimentos Específicos	Ferramentas/Exemplos
Sistemas Operacionais	Interface visual e linha de comando (CLI)	Windows, Linux, macOS
Pacote Office	Word, Excel	Google Docs, Microsoft 365
Idiomas	Inglês (essencial), Espanhol, Mandarim	-
Internet	Pesquisa avançada, manutenção de contatos	Google, LinkedIn
Segurança	Privacidade, vulnerabilidades	LGPD, OWASP
Lógica e Programação	Base lógica, linguagens de programação	Python, Java, JavaScript

Infraestrutura	Virtualização (VMs, Containers), monitoramento	Docker, VMWare, Grafana (CPU/RAM/discos)
Banco de Dados	Bancos relacionais e não relacionais	MySQL, MongoDB
Telecomunicações	Protocolos, redes	TCP/IP, HTTP/HTTPS
Ferramentas QA	**Automatização:**	Selenium, Cypress
	Performance:	JMeter
	APIs:	Postman
	Monitoramento:	Grafana, Kibana
Conhecimento do Negócio	Setores (Bancos, Seguradoras, Indústrias)	LGPD, regulamentações setoriais
Processos de Teste	Planejar → Analisar → Modelar → Executar	Priorização de ferramentas conforme contexto

⚠️ Débito Técnico

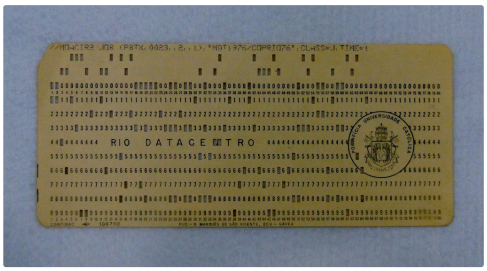
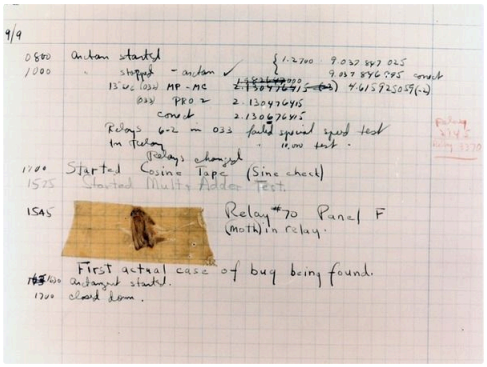
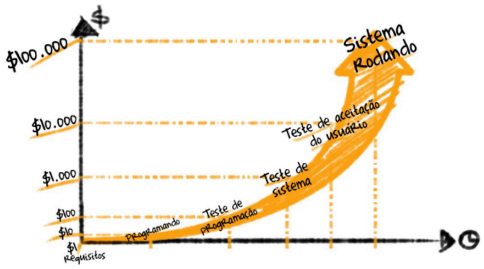
- Presente em todos os projetos; deve ser **gerenciado**, não ignorado.
- **Passos para lidar com débitos:**
  a. **Identificar** (registrar no backlog).
  b. **Priorizar** (impacto vs. urgência).
  c. **Planejar** ações de correção.
  d. **Revisar** periodicamente.
- **Exemplos comuns de débitos técnicos:**
  - Código **não** otimizado.
  - Funcionalidades **parcialmente** implementadas.
  - Documentação **incompleta** de dependências.

📌 Seção 02: Introdução ao Teste de Software (Dia 03)

- 🗓️ 26 de mar. de 2025

🕰️ Uma Breve História do Teste

Evento/Personagem	Contribuição
Ada Lovelace (1843)	Primeira programadora/testadora — identificou um erro na Máquina Analítica.

Cartões Perfurados (século XIX)	 Origem do termo *"Teste de Caixa Branca"* (armazenados em caixas brancas).
Grace Hopper (1947)	 Registrou o primeiro *"bug"* (mariposa presa em um relé).
Glenford Myers (1979)	Autor de *"The Art of Software Testing"* (bíblia dos testes).
Regra dos 10 de Myers	 *"Defeitos cobram juros: quanto mais tarde são encontrados, mais caro fica."*

💣 A Importância do Teste x Danos dos Bugs

- **Impactos:**
  - **Financeiros/Imagem:** Perda de vendas, confiança e atrasos.
  - **Governo:** Vulnerabilidades, decisões erradas (ex.: derrotas militares).
  - **Pessoas:** Risco de vida, supressão de direitos (ex.: falha em sistemas médicos).
  - **Meio Ambiente:** Poluição, desperdício de recursos (ex.: vazamentos não detectados).

📜 Os 7 Fundamentos do Teste (ISTQB)

1. **Teste mostra defeitos, mas não sua ausência** 🎯
   - Nunca garantirá 100% de ausência de bugs (exceto em sistemas críticos, como aviação).
2. **Teste exaustivo é impossível** 🔍
   - Foque em riscos e prioridades.
3. **Teste antecipado reduz custos** ⏳
   - Quanto antes o teste começar, mais barato será corrigir defeitos.
4. **Agrupamento de defeitos** 🎲
   - 80% dos bugs estão em 20% dos módulos (*Princípio de Pareto*).

5. **Paradoxo do pesticida** 🐞
    - Testes repetitivos perdem eficácia; atualize casos de teste frequentemente.
6. **Teste depende do contexto** 🏗️
    - Um sistema de aviação ≠ um site de e-commerce (riscos diferentes).
7. **Ilusão da ausência de erros** ❌
    - Software “perfeito” pode não atender às necessidades do usuário (ex.: Windows X OS/2).

🔄 Diferença entre Teste e QA

Aspecto	Teste de Software	Garantia da Qualidade (QA)
Foco	Produto (funcionalidade, desempenho, bugs)	Processo (metodologias, melhorias contínuas)
Objetivo	Identificar e corrigir defeitos	Prevenir defeitos e aprimorar processos
Abordagem	Detectar falhas após o desenvolvimento	Criar estratégias para evitar falhas desde o início
Atuação	Testa o software em busca de erros	Define padrões e práticas para garantir qualidade
Responsabilidade	Equipe de Testes ou QA	Todos os envolvidos no desenvolvimento
Atividades	Execução de testes manuais e automatizados, análise de defeitos	Definição de processos, revisão de código, treinamento, monitoramento de qualidade
Relacionamento com Testes	Parte do processo de QA	Engloba testes e outras práticas de qualidade
Melhoria Contínua	Corrige problemas conforme encontrados	Aprende com erros passados para evitar reincidência
Prevenção x Detecção	Detecção de defeitos	Prevenção de defeitos

🔍 Erro, Ocorrência, Defeito e Falha

Termo	Definição	Origem	Exemplo	Observações Chave
Erro (Engano)	Ação humana que produz um resultado incorreto.	Pessoa (dev, analista, etc.)	Um desenvolvedor esquece de validar um campo de e-mail no formulário.	• Pode ocorrer em qualquer fase (requisitos, código, documentação). • É a causa raiz.
Defeito (Bug)	Manifestação do erro no produto (código,	Resultado do erro	O campo de e-mail aceita caracteres	• Pode existir sem ser executado.

	documento ou sistema).		inválidos como "*!;\=@".	• Chamado de "defeito" quando identificado por terceiros.
Ocorrência	Suspeita de um comportamento anômalo (ainda não confirmado como defeito).	Durante testes ou uso	Um tester observa que o sistema travou ao digitar "@" no campo e-mail.	• Estado intermediário entre observação e confirmação. • Requer investigação.
Falha	Comportamento visível do sistema quando um defeito é executado.	Execução do código com defeito	O usuário digita "nome@" e o sistema crasha.	• Só ocorre se o defeito for ativado. • Impacta diretamente o usuário final.

1. **Diferença entre "erro" e "defeito":**
   - Se **eu** cometo um equívoco: **Erro**.
   - Se **outra pessoa** identifica meu equívoco: **Defeito**.
2. **Hierarquia de Causas:**

```
Erro Humano → Defeito no Código → Falha em Execução
```

3. **Segundo o ISTQB:**
   - Defeito = Anomalia identificada **sem execução** (ex.: revisão de código).
   - Falha = Defeito **ativado durante execução**.
4. **Casos Especiais:**
   - **Defeitos que não causam falhas:**
     - Código inacessível (ex.: função não chamada).
     - Condições muito específicas não testadas.
5. **Impacto nos Testes:**
   - **Ocorrências** devem ser **investigadas** antes de virar defeitos reportados.
   - **Falhas** priorizadas por criticidade (ex.: crash > erro visual).

📊 Tipos de Testes Baseados na IEC/ISO 25010 (SQuaRE - AFUCCEMPS)

Característica (Sigla)	Subcaracterísticas	O Que Avaliar	Exemplo Prático
Adequação Funcional (A)	Completude Funcional Correção Funcional Apropriação Funcional	Se o sistema faz o que deveria fazer	Verificar se um caixa eletrônico realiza saques corretamente

Usabilidade (F)	Reconhecibilidade Aprendibilidade Operabilidade Proteção contra erros Estética UI Acessibilidade	Facilidade de uso e aprendizado	Testar se um idoso consegue usar um aplicativo de banco
Compatibilidade (U)	Coexistência Interoperabilidade	Funciona com outros sistemas	Verificar integração entre ERP e sistema de estoque
Confiabilidade (C)	Maturidade Disponibilidade Tolerância a falhas Recuperabilidade	Resiste e se recupera de falhas	Testar comportamento após queda de energia
Eficiência (C)	Comportamento temporal Utilização de recursos Capacidade	Desempenho e consumo de recursos	Medir tempo de resposta com 1000 usuários simultâneos
Manutenibilidade (E)	Modularidade Reusabilidade Analisabilidade Modificabilidade Testabilidade	Facilidade de manutenção	Avaliar tempo para corrigir um bug reportado
Portabilidade (M)	Adaptabilidade Instalabilidade Substituibilidade	Funciona em diferentes ambientes	Testar em Windows, Linux e macOS
Segurança (P)	Confidencialidade Integridade Não repúdio Responsabilidade Autenticidade	Proteção de dados e acesso	Testar resistência a SQL Injection

🤖 Testes Manuais X Automatizados

Critério	Testes Manuais	Testes Automatizados
Execução	Realizado por humanos (passo a passo).	Executado por scripts/ferramentas.
Melhor uso	• Novas funcionalidades • Casos complexos/subjetivos • UX/UI	• Regressão • Testes repetitivos • Carga/Performance
Velocidade	Mais lento (depende do tester).	Rápido (execução paralela e contínua).

Custo inicial	Baixo (nenhuma infraestrutura complexa).	Alto (setup de ferramentas + manutenção).
Flexibilidade	Adaptável a mudanças rápidas.	Requer ajustes nos scripts para mudanças.
Cobertura	Limitada (tempo humano finito).	Ampliada (executa milhares de casos em minutos).
Exemplos	• Exploratórios • Usabilidade • Protótipos	• API (Postman) • E2E (Selenium) • Performance (JMeter)

1. **Integração Contínua:**
   - Automação é essencial para **pipelines de CI/CD** (testes rodam a cada commit).
2. **Foco do Tester Manual:**
   - Pode se dedicar a:
     - **Novas funcionalidades** (não automatizadas ainda).
     - **Testes exploratórios** (bugs inesperados).
3. **Ferramentas Comuns:**
   - **Automação:** Selenium (Web), Appium (Mobile), RestAssured (API).
   - **Manuais:** Checklists, sessões exploratórias.

🎯 Quando Usar Cada Abordagem?

Cenário	Melhor Escolha	Por quê?
Lançamento de feature nova funcionalidade	Manual + Automatizado	Validação rápida (manual) + Regressão (auto).
Testes de usabilidade	Manual	Requer avaliação humana de UX.
Builds frequentes	Automatizado	Agilidade na execução.

🔄 Testes Tradicionais X Ágeis

- Os testes tradicionais e os testes ágeis são abordagens diferentes para testar software. Os testes tradicionais são mais lineares e sequenciais, enquanto os testes ágeis são mais iterativos e incrementais.

Tradicional (Cascata)	**Ágil** (Scrum/Kanban)
Teste após desenvolvimento.	Teste em paralelo com desenvolvimento.
Documentação extensa.	Foco em feedback rápido e iterativo.
Menos flexível a mudanças.	Adaptável a requisitos dinâmicos.

📌 Seção 03: Atitudes de um Profissional da Qualidade (Dia 04)

- 🗓️ 27 de mar. de 2025

⚡ Pressão Organizacional

- **Quem pressiona o QA:**
  - Desenvolvedores, POs, Scrum Masters, gestores (de produto, projeto, infra), clientes e usuários.
- **Seu papel como "Conselheiro do Rei":**
  - **Lembrar** a equipe sobre:
    - **Qualidade Mínima Viável (MVQ)**
    - **Riscos do não testado** ("Bugs cobram juros")
    - **Lições de decisões passadas**
  - **Firmeza:** Defender a qualidade mesmo sob pressão por prazos.

🔄 Comprometido vs. Envolvido

Envolvido	Comprometido
Quer emprego, não responsabilidade.	Faz por paixão e busca excelência.
Passivo ("Cada um no seu quadrado").	Ativo ("Visão de dono").
Age por "jeitinho".	Meritocrata (valoriza conquistas e aprendizado).

> ***"Comprometido encontra algo que gosta: faz bem, é reconhecido e bem remunerado."***
>
> ***"Meritocracia não é ingênua – empresas mudam, mas suas habilidades ficam."***

⏳ Autogerenciamento

- **Recursos limitados:** Tempo, energia, tarefas (importantes/urgentes), restrições.
- **Como gerenciar:**
  a. **Classificar tarefas** com:
    - **Priorização** (Pareto: 20% esforço → 80% resultados).
    - **GUT** (Grave, Urgente, Tendencioso).
  b. **Negar/delegar** o não essencial.
  c. **Medir** tempo vs. resultado.

> ***"Não se troca presença por presentes – tempo é finito."***

📊 Técnica GUT para Priorização em QA

Critério	O Que Significa	Exemplo em Testes/QA	Nível de Prioridade
Grave	Impacto alto no sistema ou usuário.	• Bug que corrompe dados do banco. • Falha de segurança que expõe informações sensíveis.	🔴 **Máxima** (Resolver imediatamente)
Urgente	Prazos curtos ou consequências imediatas.	• Bug crítico em produção antes do lançamento. • Testes pendentes para entrega em 24h.	🟠 **Alta** (Resolver rápido)
Tendencioso	Pode piorar ou gerar novos	• Defeito pequeno que, se ignorado, pode travar o sistema no futuro.	🟡 **Média** (Planejar correção)

	problemas se não tratado.	• Falha de design que acumulará débitos técnicos.	

💬 Comunicação Eficaz

- **Componentes da Mensagem**

Elemento	Influência	Exemplos Práticos em QA	Dicas
Palavras (10%)	Conteúdo literal.	• Relatórios de bugs. • E-mails formais ao time.	Evite gírias; seja técnico, mas claro.
Tom de Voz (30%)	Como se fala.	• Reuniões diárias (Scrum). • Feedback a devs.	Mantenha calma mesmo sob pressão.
Não Verbal (55%)	Linguagem corporal.	• Postura em reuniões. • Expressões ao demonstrar um bug.	Contato visual e gestos abertos transmitem confiança.

- **Tipos de Comunicação**

🔹 **Verbal**

Tipo	Uso em QA	Cuidados
Escrita	• Documentar casos de teste. • Reportar bugs em ferramentas (Jira).	• Escrever como profissional (não como mensagem casual). • Incluir screenshots e passos reproduzíveis.
Oral	• Daily meetings. • Explicar riscos ao PO.	• Evitar monólogos; fazer perguntas para engajar.

🔹 **Não Verbal**

Categoria	Impacto em QA	Exemplo
Símbolos	Ícones em dashboards de qualidade.	🚨 (Urgente), ✅ (Aprovado).
Aparência	Credibilidade em reuniões.	Vestir-se adequadamente ao ambiente (presencial/remoto).
Cinésica	Gestos durante demonstrações.	Apontar para telas em testes exploratórios.
Proxêmica	Distância em pair testing.	Não invadir espaço pessoal do dev durante colaboração.

- **Barreiras Comuns em QA**

Barreira	Solução	Exemplo
Termos técnicos	Adaptar linguagem ao público.	Explicar um "false positive" ao PO como "alarme falso".
Culturais	Pesquisar gestos locais.	Em alguns países, acenar com a cabeça ≠ concordância.
Ambiente ruidoso	Usar ferramentas claras (Slack, Zoom).	Enviar resumo por escrito após reuniões caóticas.

- **Comunicação em Crises**
  - **Quando reportar um bug crítico:**
    - **Contexto rápido:** "Encontrei um bloqueador na tela de pagamentos (GUT: Grave=5, Urgente=5)".
    - **Evidências:** Screenshot + logs anexados.
    - **Sugestão:** "Sugiro rollback da versão até a correção."

> *"**Às vezes as pessoas não te ouvem porque você não dá ênfase ao que diz – use os 55% não verbais a seu favor."***

Exemplo:

- Ao explicar um risco complexo:
  - ❌ Falar baixo e olhar para o notebook.
  - ✅ Postura ereta, gestos claros e tom firme.

🤝 Negociação (Ganha-Ganha) – Para QAs

Conceito:

- Buscar soluções onde **todos ganham** (equipe, clientes, produto).

Como aplicar em QA:

Tipo	Cenário Comum	Estratégia Ganha-Ganha
Prazos vs. Qualidade	Time pressionado para entregar sem testes completos.	Negociar: "Reduzimos escopo de testes, mas incluímos checks automáticos críticos."
Priorização de Bugs	Devs alegam que "não é bug, é feature".	Usar critérios objetivos (ex.: impacto no usuário + evidências).

> ***"Negociações falhas são Perde-Perder – ex.: liberar com bugs = insatisfação do cliente + retrabalho para o time."***

🚀 Produtividade – Métodos para QAs

Hábitos Eficientes:

- **Foco na jornada:** "Curtir o processo, não só o resultado."
- **Gratidão:** Anotar 1 conquista diária (ex.: bug crítico encontrado).

Técnicas Comprovadas:

Método	Como Usar em QA	Exemplo

Timeboxing	Reservar blocos para tarefas específicas.	"Das 10h às 12h: testar fluxo de checkout."
Regra dos 2 Minutos	Se uma tarefa leva ≤ 5 minutos, faça agora.	Responder a um e-mail de confirmação de bug.

> ***"Sua produtividade não é medida por horas trabalhadas, mas por impacto gerado."***

🔄 Fluxo Contínuo (Kanban para QAs)

Princípios:

- **Limite de trabalho em progresso (WIP):** Máx. 3 tarefas "Em Teste" por vez.
- **Colunas essenciais:**

```
[A Fazer] → [Em Construção] → [Em Teste] → [Em Implantação] → [Feito/Pronto]
```

Onde as colunas “Em Teste” e “Em Implantação”, são “opcionais”, mas desejaveis para assim manter uma melhor qualidade de software.

> ***"Kanban não é só quadro – é sobre ritmo sustentável."***

⏳ Técnica Pomodoro

Passo a Passo:

1. **Selecione** 1 tarefa (ex.: testar login mobile).
2. **25min focado:**
   - Navegação + registro de bugs (sem distrações).
3. **5min de pausa:**
   - Alongar, beber água.
4. **Após 4 ciclos:**
   - Pausa longa (15-30min) para relatório consolidado.

Benefícios em QA:

- Evita burnout em testes repetitivos.
- Aumenta cobertura (foco total em cada sessão).

"Pomodoro é ‘devagar se vai ao longe’ – 25min de teste exploratório podem achar mais bugs que 2h distraído."

🔎 Insights e Observações Gerais

- O QA é um **agente de prevenção**, não apenas de correção.
- **Soft skills** são tão importantes quanto habilidades técnicas.
- **Débito técnico** deve ser tratado de forma estratégica, não como “problema futuro”.
- **“Bugs são como juros”:** Corrija-os cedo para evitar custos exponenciais (Myers).
- **QA ≠ Testador:** QA é estratégico, testador é tático.
- **Automatizar é econômico,** mas não substitui o pensamento crítico humano.
- **Qualidade exige resiliência:** Mantenha-se firme sob pressão.
- **Comprometimento > Envolvimento:** Faça a diferença, não apenas “apareça”.
- **Comunicação não verbal importa mais que palavras.**
- **Produtividade sustentável** > Correria desorganizada.