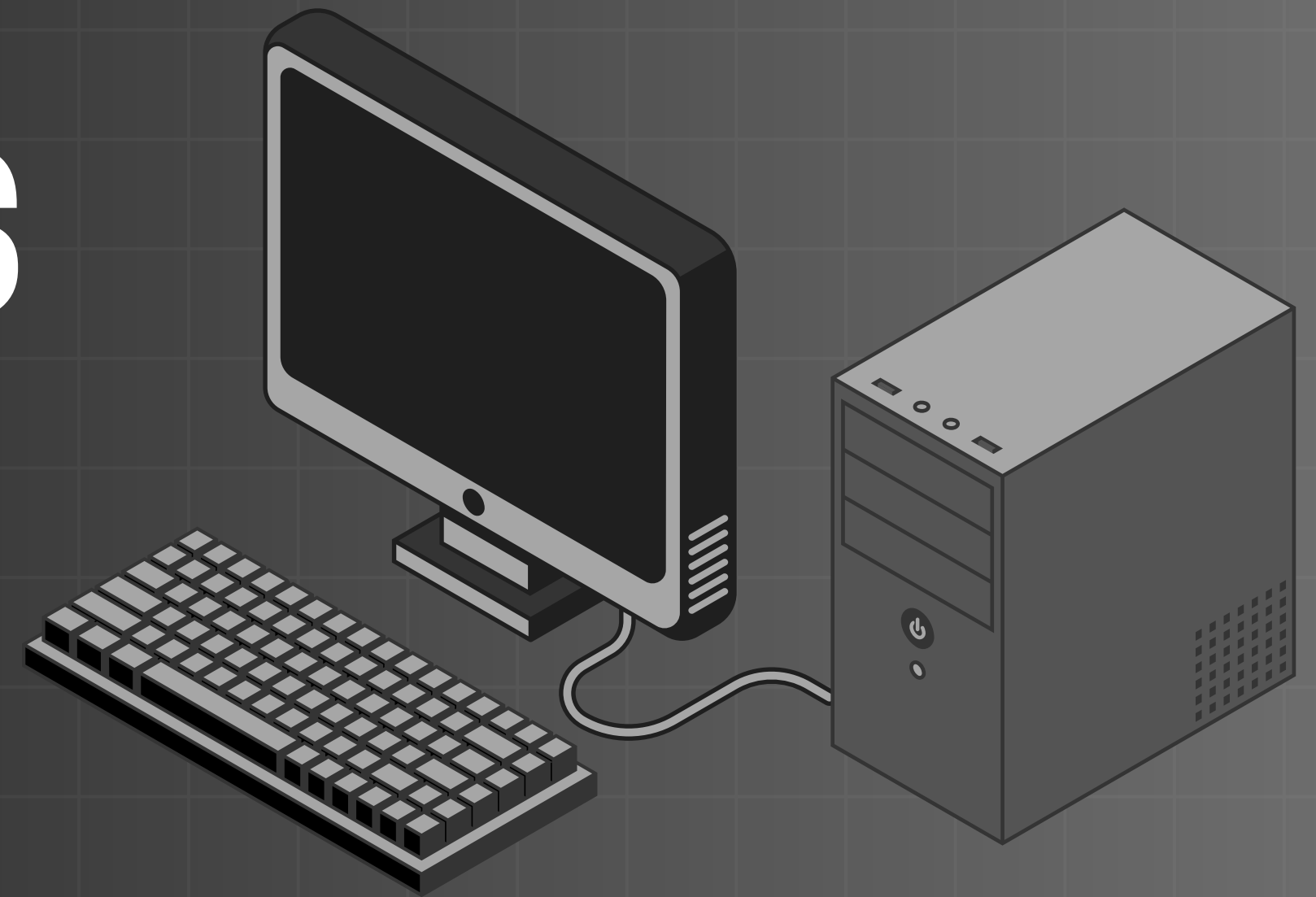
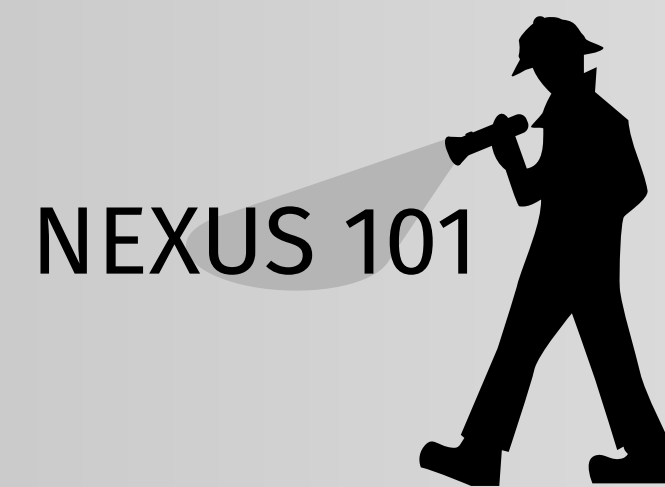


FERRAMENTAS DE TESTE

Apresentar as principais ferramentas de teste de software utilizadas na indústria, destacando suas funcionalidades, diferenciais e importância no ciclo de desenvolvimento.





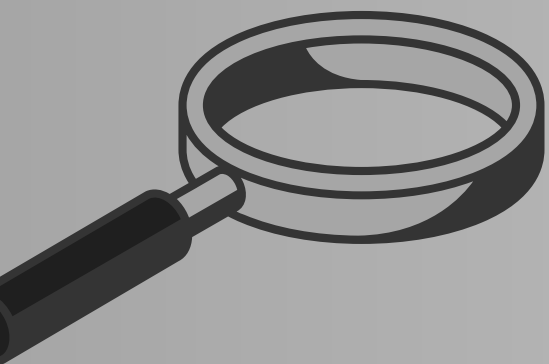
KAREN KÉSSIA HERRMANN

WESLEY CASSIO

MARIA EDUARDA

ANNA BEATRIZ SANTORO

DOUGLAS PAULO CORTES



Por que utilizar ferramentas de teste?



Automatizam tarefas repetitivas



Reduzem erros humanos



Aumentam a eficiência



Melhoram a rastreabilidade e organização

SUPOORTE DE FERRAMENTAS PARA TESTES



CLASSIFICAÇÃO DAS FERRAMENTAS DE TESTE

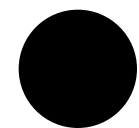
Categoria	Finalidade
• Ferramentas de gerenciamento	Aumentam a eficiência do processo de teste, facilitando o gerenciamento do SDLC, dos requisitos, dos testes, dos defeitos e da configuração
• Ferramentas de teste estático	Dão suporte ao Testador na realização de revisões e análises estáticas
• Ferramentas de projeto e implementação de testes	Facilitam a geração de casos de teste, dados de teste e procedimentos de teste
• Ferramentas de execução e cobertura de testes	Facilitam a execução de testes automatizados e a medição da cobertura
• Ferramentas de teste não funcional	Permitem que o Testador realize testes não funcionais que são difíceis ou impossíveis de serem realizados manualmente
• Ferramentas de DevOps	Suporte ao pipeline de entrega de DevOps, rastreamento de fluxo de trabalho, processo(s) de construção automatizado(s), CI/CD

CLASSIFICAÇÃO DAS FERRAMENTAS DE TESTE

Categoria	Finalidade
• Ferramentas de colaboração	Facilitam a comunicação
• Ferramentas que oferecem suporte à escalabilidade e à padronização da implantação	(Ex: máquinas virtuais, ferramentas de containerização)
• Qualquer outra ferramenta que auxilie no teste	(Ex: uma planilha pode ser uma ferramenta)

PRINCIPAIS

FERRAMENTAS POR CATEGORIA



JIRA

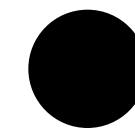
Gerenciamento

- Linguagens compatíveis: Qualquer linguagem (Java, Python, etc.)
- Diferenciais: Foco em agilidade, personalizável e com ótimas integrações.
- Use quando: Gerenciar tarefas, bugs ou sprints.
- Curiosidade: “Jira” vem de “Gojira” — o nome japonês do Godzilla.

SONARQUBE

Testes Estáticos

- Linguagens compatíveis: Java, JavaScript, Python, C#, e mais de 25 outras.
- Diferenciais: Analisa código estático, detecta bugs, vulnerabilidades e code smells automaticamente.
- Use quando: Avaliar a qualidade e manter padrões de desenvolvimento.
- Curiosidade: O nome “Sonar” vem da ideia de escanear o código.



WORD

Testes Estáticos

- Linguagens compatíveis: Documentos textuais.
- Diferenciais: Suporte a revisões, comentários e controle de alterações.
- Use quando: Revisar documentos e código em formato textual.
- Curiosidade: É amplamente utilizada por equipes de QA para testes estáticos formais e informais.

PRINCIPAIS

FERRAMENTAS POR CATEGORIA

TESTLINK

Projeto e Implementação de Testes

- Linguagens compatíveis: Qualquer linguagem (gerencia testes, não executa código).
- Diferenciais: Open source, organiza casos e execuções de teste com rastreabilidade.
- Use quando: Precisa documentar e acompanhar testes manuais.
- Curiosidade: Ainda é popular por ser gratuito e direto ao ponto.

SELENIUM

Execução e cobertura de testes

- Linguagens compatíveis: Java, Python, JavaScript, C#, Ruby, entre outras.
- Diferenciais: Automação de testes em navegadores reais, open source.
- Use quando: Automatizar testes funcionais de aplicações web.
- Curiosidade: Seu nome referencia ao ao selênio, usado como “antídoto” ao Mercury, uma antiga ferramenta rival.

MITRE ATT&CK

Teste não funcional

- Linguagens compatíveis: Não se aplica (base de conhecimento).
- Diferenciais: Mapeia táticas e técnicas de ataque reais.
- Use quando: Planejar testes de segurança e avaliar ameaças.
- Curiosidade: “ATT&CK” = Adversarial Tactics, Techniques & Common Knowledge.

PRINCIPAIS

FERRAMENTAS POR CATEGORIA

GITHUB ACTIONS

DevOps

- Linguagens compatíveis: Qualquer linguagem hospedada no GitHub.
- Diferenciais: Automatiza fluxos CI/CD diretamente no repositório.
- Use quando: Executar testes, builds ou deploys a cada push ou PR.
- Curiosidade: Cada “action” é um passo reutilizável — você pode até criar as suas!

JENKINS

DevOps

- Linguagens compatíveis: Qualquer linguagem.
- Diferenciais: Altamente extensível, com centenas de plugins para CI/CD.
- Use quando: Automatizar builds, testes e deploys em pipelines personalizados.
- Curiosidade: É um dos pioneiros em integração contínua e ainda muito usado em grandes projetos.

BITBUCKET

DevOps

- Linguagens compatíveis: Qualquer uma — é um repositório Git.
- Diferenciais: Integração nativa com Jira, suporte a pipelines CI/CD e controle de acesso detalhado.
- Use quando: Hospedar código, revisar PRs e automatizar testes/deployes.
- Curiosidade: Criado pela Atlassian, é a alternativa natural ao GitHub para times que já usam Jira e Confluence.

PRINCIPAIS

FERRAMENTAS POR CATEGORIA

TEAMS

• Colaboração

- Linguagens compatíveis: Todas — foco em colaboração.
- Diferenciais: Chats, reuniões, arquivos e integrações no ecossistema Microsoft 365.
- Use quando: Comunicar, compartilhar e colaborar em tempo real com o time.
- Curiosidade: Cresceu rapidamente como resposta ao Slack, com forte presença em empresas que usam Office.

NOTION

Colaboração

- Não se aplica (organização e documentação).
- Diferenciais: Tudo-em-um: notas, wikis, tarefas e páginas personalizáveis.
- Use quando: Centralizar informações, organizar testes e fluxos.
- Curiosidade: Popular entre devs pela flexibilidade e visual limpo.

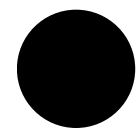
SLACK

Colaboração

- Linguagens compatíveis: Todas — comunicação, não codificação.
- Diferenciais: Mensagens em tempo real com canais, integrações e bots.
- Use quando: Facilitar a comunicação do time e integrar alertas (builds, testes, deploys).
- Curiosidade: O nome vem de “Searchable Log of All Conversation and Knowledge”.

PRINCIPAIS

FERRAMENTAS POR CATEGORIA



CONFLUENCE

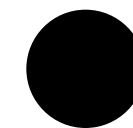
Colaboração

- Linguagens compatíveis: Não se aplica (documentação).
- Diferenciais: Criação colaborativa de documentos, integração com Jira.
- Use quando: Documentar requisitos, testes e processos.
- Curiosidade: Favorece times ágeis com documentação viva e centralizada.

DOCKER

Oferecem suporte à escalabilidade e à padronização da implantação

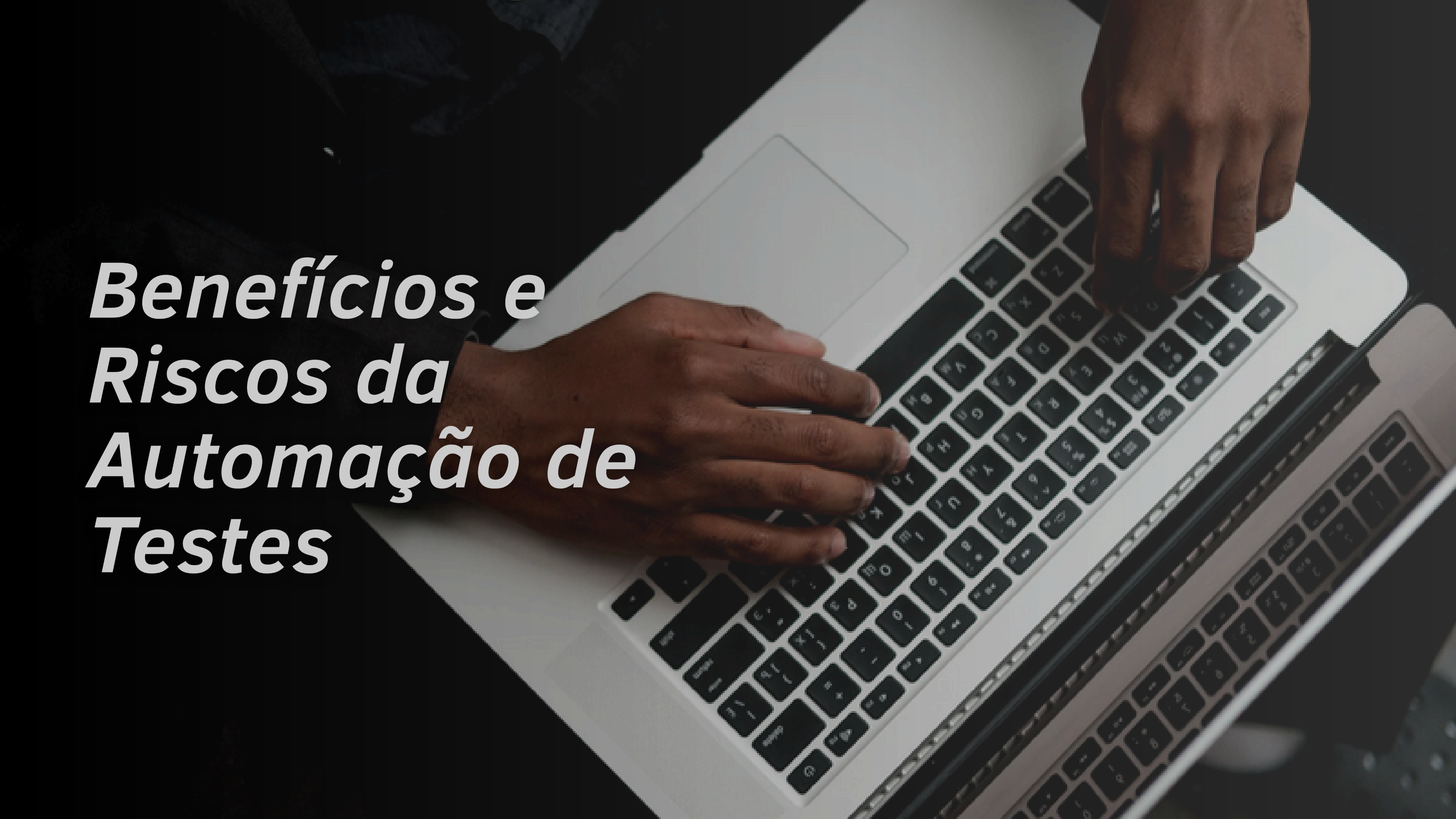
- Linguagens compatíveis: Todas.
- Diferenciais: Cria containers leves e portáteis.
- Use quando: Padronizar ambientes e automatizar testes/deloys.
- Curiosidade: O mascote é uma baleia chamada Moby Dock, carregando containers.



EXCEL

Qualquer outra ferramenta que auxilie no teste

- Linguagens compatíveis: CSV.
- Diferenciais: Organiza dados, gera gráficos, aplica fórmulas e macros.
- Use quando: Controlar casos de teste, registrar resultados ou gerar relatórios simples.
- Curiosidade: Apesar de básico, ainda é muito usado como ferramenta de apoio em QA.

A close-up, high-angle photograph of a person's hands typing on a silver laptop keyboard. The hands are dark-skinned. The laptop is open, and the keyboard is clearly visible. The background is dark and out of focus.

Benefícios e Riscos da Automação de Testes

QUAIS OS BENEFÍCIOS DAS FERRAMENTAS DE TESTE?

01

Aceleram o processo de validação do software.

02

Reduzem retrabalho e custos com correções tardias.

03

Aumentam a confiança na entrega.

04

Viabilizam testes em escala, com qualidade e rastreabilidade.

QUAIS OS BENEFÍCIOS DAS FERRAMENTAS DE TESTE?

05

Redução no trabalho
manual repetitivo

06

Maior consistência e
repetibilidade

07

Avaliação mais objetiva

08

Acesso mais fácil a
informações sobre testes

QUAIS OS BENEFÍCIOS DAS FERRAMENTAS DE TESTE?

09

Redução dos tempos de
execução

10

Mais tempo para os
Testadores criarem testes
novos

11

Decisões mais rápidas e
baseadas em dados

12

Relatórios automatizados e
acesso fácil a estatísticas

RISCOS

EXPECTATIVAS IRREAIS
SOBRE OS BENEFÍCIOS
DE UMA FERRAMENTA

A DEPENDÊNCIA DO
FORNECEDOR DA
FERRAMENTA

ESTIMATIVAS IMPRECISAS

USAR UM SOFTWARE
DE CÓDIGO ABERTO
QUE PODE SER
ABANDONADO

RISCOS

A FERRAMENTA DE
AUTOMAÇÃO NÃO SER
COMPATÍVEL

USAR UMA
FERRAMENTA DE
TESTE QUANDO O
TESTE MANUAL É MAIS
APROPRIADO

ESCOLHA DE UMA
FERRAMENTA
INADEQUADA

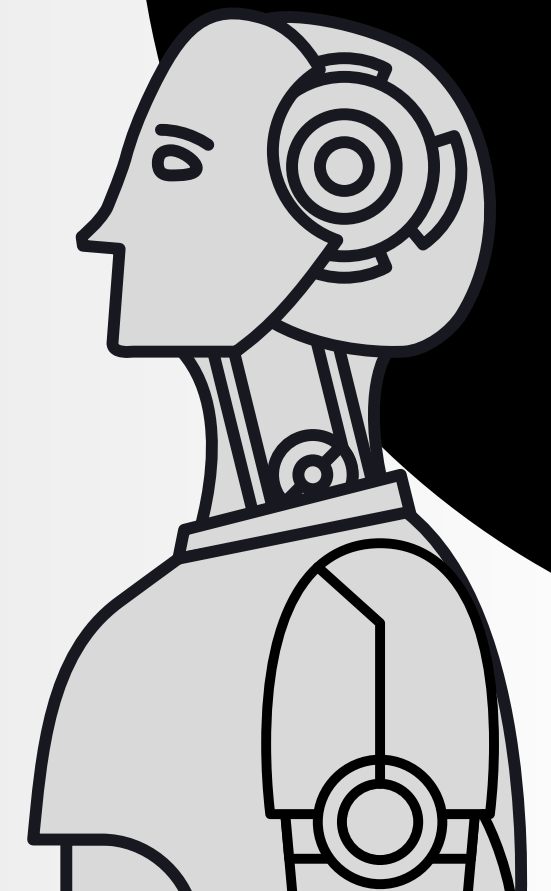
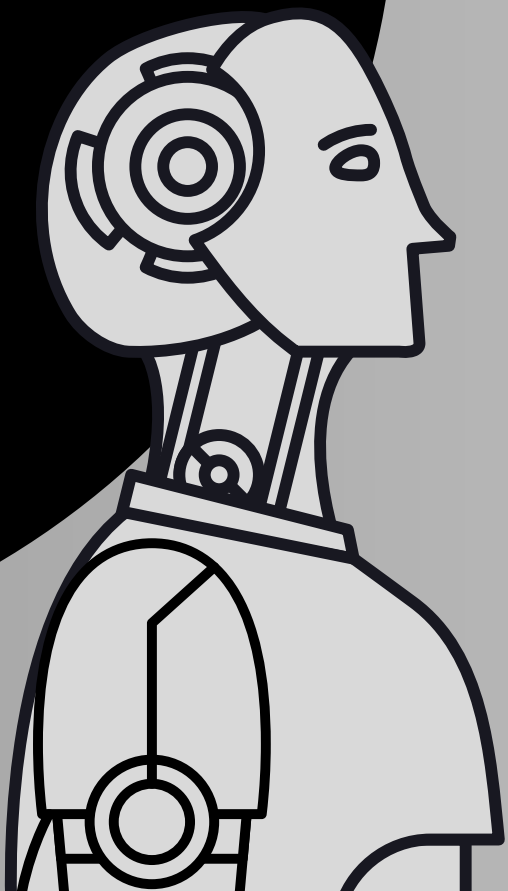
CONFIAR DEMAIS EM
UMA FERRAMENTA

TENDÊNCIAS E FUTURO

DAS FERRAMENTAS DE TESTE

1. Testes com IA (Inteligência Artificial): geração automática de casos de teste.
2. Testes baseados em código (TestOps): união entre DevOps e QA.
3. Low-code/No-code testing: democratização do teste.

Exemplo: Testim e mabl usam IA para sugerir testes automaticamente.





INTEGRAÇÃO INTELIGENTE DE FERRAMENTAS: O CAMINHO PARA TESTES EFICAZES

FERRAMENTAS SÃO ALIADAS ESTRATÉGICAS. QUANDO BEM UTILIZADAS, TRANSFORMAM TESTES EM UM DIFERENCIAL COMPETITIVO.

AUTOMATIZAM, ANALISAM, DOCUMENTAM, INTEGRAM E COLABORAM

POTENCIALIZAM O TRABALHO HUMANO AO INVÉS DE SUBSTITUÍ-LO

TORNAM O PROCESSO DE TESTES MAIS ÁGIL, PADRONIZADO E CONFIÁVEL



Suporte de Ferramentas para Testes

Benefícios da Automação de Testes

- Redução do trabalho manual repetitivo
- Maior consistência e repetibilidade
- Avaliação mais objetiva
- Acesso mais fácil a informações sobre testes
- Redução dos tempos de execução de testes
- Mais tempo para criar novos testes

Riscos da Automação de Testes

- Expectativas irreais sobre os benefícios da ferramenta
- Estimativas imprecisas de tempo, custos e esforço
- Uso inadequado de ferramentas quando o teste manual é mais apropriado
- Dependência excessiva da ferramenta
- Dependência do fornecedor da ferramenta
- Uso de software de código aberto que pode ser abandonado
- Incompatibilidade da ferramenta com a plataforma de desenvolvimento
- Escolha de ferramenta inadequada para requisitos normativos e de segurança

Nexus 101

Obrigado!

Qualquer dúvida chamem no privado!