ATIVIDADE: HISTÓRICO E EVOLUÇÃO

1. Quais foram as primeiras iniciativas para a criação de testes de software?

Anos 1950–1960: os primeiros testes eram manuais e focados em depuração (testar era "caçar bugs").

- Alan Turing (1949) já mencionava a necessidade de verificar programas, propondo a "verificação automática".
- Glenford Myers (1979) com o livro *The Art of Software Testing* foi um marco, pois consolidou o teste como atividade distinta da programação, defendendo que "testar não é provar que funciona, mas sim encontrar falhas".
- Década de 1970–1980: surgiram métodos formais (*caixa-preta* e *caixa-branca*), além da separação entre teste de unidade, integração e sistema.

2. Como são feitos testes no Google?

Segundo James A. Whittaker, Jason Arbon e Jeff Carollo no livro *How Google Tests Software* (2012):

- Papel do Engenheiro de Teste: em vez de testadores manuais isolados, todos os engenheiros são responsáveis pela qualidade, apoiados por Test Engineers (TEs).
- Automação em larga escala: a maior parte dos testes é automatizada (unitários, integração, regressão, performance).
- Testes em produção: uso de canary releases, *A/B testing* e monitoramento contínuo.
- Infraestrutura massiva: execução de milhões de testes diariamente em sistemas distribuídos.
- Cultura de qualidade: "testar é responsabilidade de todos" (*shift-left testing*).

3. Cite um autor de referência sobre o tema"Testes"? O que deve haver em um plano de testes?

- Glenford J. Myers – *The Art of Software Testing* (1979).
- Outros autores importantes:
    - Boris Beizer (*Software Testing Techniques*, 1983).

ATIVIDADE: HISTÓRICO E EVOLUÇÃO

- Cem Kaner (*Testing Computer Software*, 1999).
- Rex Black (*Managing the Testing Process*, 2002).

4. O que deve haver em um Plano de Testes

Um plano de testes (segundo ISTQB, IEEE 829 e ISO/IEC/IEEE 29119) deve conter:

1. Identificação: título, versão, responsáveis.
2. Introdução: objetivos e escopo do plano.
3. Itens de teste: software/módulos a serem testados.
4. Funcionalidades a testar: requisitos, casos de uso, fluxos.
5. Funcionalidades que não serão testadas (fora de escopo).
6. Estratégia e abordagem: tipos de testes (unitário, integração, sistema, regressão, desempenho etc.).
7. Critérios de entrada e saída: condições para iniciar e encerrar os testes.
8. Recursos necessários: equipe, ferramentas, ambientes.
9. Cronograma: prazos e marcos importantes.
10. Riscos e contingências: possíveis problemas e como mitigar.
11. Artefatos: casos de teste, scripts, dados de teste, relatórios.