• CRITÉRIOS PARA O AUTOR:

↳ Seja seu primeiro revisor

↳ Entenda que a revisão não mede ou avalia a sua competência

↳ Defina PRs pequenos para facilitar a revisão (no máx. 200 linhas de códigos)

↳ PRs pequenos ≠ PRs rascunho

- projetos inacabados
- versões rudimentares do projeto //

↳ Dicas de nomeação de Pull Requests:

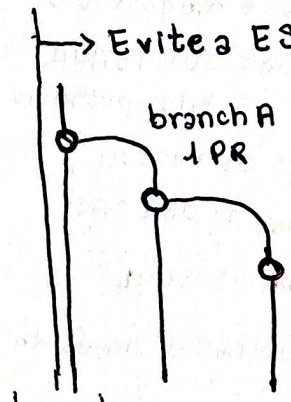
>> PREFIXOS:

- Feat (novas funcionalidades)
- Fix (correção de uma feature existente)
- chore (não impacta a percepção do usuário)

>> PRs CONCISOS E LINKADOS

>> SQUASH - alinhamento e DESCRIÇÃO de cada commit //

↳ Evite a ESCADA DE PRs



↳ Ambientes diferentes com Pull Requests diferentes //

↓

SUSCETÍVEL A CONFLITOS!

↓

EVITE!

↳ Certifique-se que a documentação associada está completa e faz parte do PR

↳ Escreva mensagens de commit significativas e claras!

• CRITÉRIOS PARA A EQUIPE:

↳ Mantenham o ritmo da equipe de acordo com o tamanho da empresa, rotatividade e confiança //

↳ Antecipem cenários de urgência sempre com senso de responsabilidade!

↳ Realizem PARES de mentoria e Code Review

Testes Automatizados

↳ Os testes automatizados são exemplos de análises dinâmicas de código, pois requerem a EXECUÇÃO contínua do sistema //

↳ Simulações e testes são exemplos de análises dinâmicas //

* Teste de Software

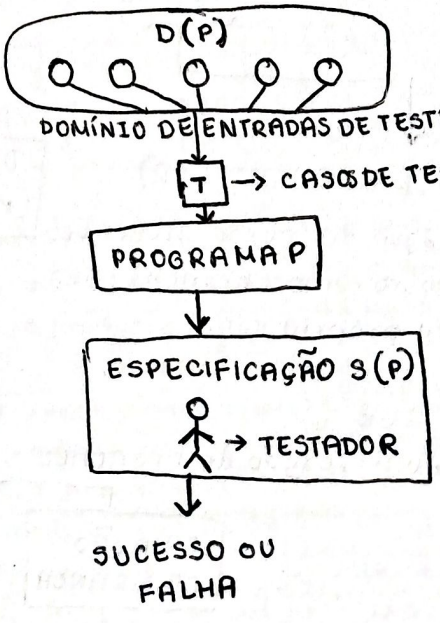
↳ Processo de executar o software de modo controlado, observando seu comportamento em relação aos requisitos especificados //

↳ As responsabilidades do teste de software não se restringem somente ao QA, mas a todos os envolvidos no workflow do projeto! (inclusive o desenvolvedor)

↳ O principal objetivo é ~~AEA~~ REVELAR A PRESENÇA DE DEFEITOS ~~EN~~ E EVENTUAIS ERROS // (com uma perspectiva de LONGO PRAZO)

↳ O teste bem sucedido é aquele que conseguem determinar os casos de teste que resultem na FALHA do programa analisado //

* Cenário de Teste



* Engano × Defeito × Erro × Falha

Engano: cometido pelo programador que introduz um → Defeito: no código que, quando ativado, pode produzir um → Erro: que, se propagado até a saída do software, constitui uma → Falha

* Níveis de Teste de Software

- TESTE UNITÁRIO
  - ↳ sub-rotina ou função
- TESTE DE INTEGRAÇÃO
  - ↳ integração de módulos
- TESTE DE SISTENA
  - ↳ toda a aplicação

} O nível de granulação é relativo ao contexto do software!

* Como Testar um Software?

↳ O teste não criterioso de um software em que todas as ~~etapas~~ entradas ~~se~~ possíveis são inseridas é chamado TESTE EXAUSTIVO e não mostra viabilidade ~~em~~ na maioria dos sistemas

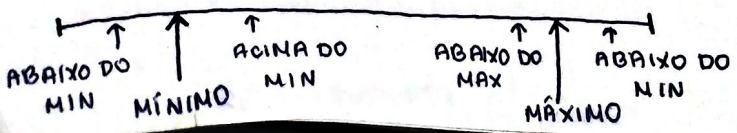
- TÉCNICAS DE TESTE - FORMAS DE PROJETAR UM CASO DE TESTE, EVITANDO ASSIN O TESTE EXAUSTIVO //
  - ↳ As principais são:

- TESTE FUNCIONAL - CAIXA PRETA
  - ↳ O teste é projetado considerando apenas as informações do próprio software sem o seu código!
  - ↳ Útil quando não há acesso à documentação do software! //

- TESTE ESTRUTURAL - CAIXA BRANCA
  - ↳ Projetado considerando a estrutura do software a partir de seu código //

- TESTE BASEADO EM DEFEITOS
  - ↳ Projetado a partir de defeitos inseridos propositalmente no código do software //

- TÉCNICA FUNCIONAL - CRITÉRIOS:
  - ↳ Particionamento em classes de equivalência (Pega uma funcionalidade e divide em classes válidas e inválidas com determinadas ENTRADAS)
  - ↳ Análise do Valor Limite



- TÉCNICA ESTRUTURAL - CRITÉRIOS:
  - ↳ Utiliza a estrutura do código para derivar casos de teste //
  - ↳ Grafo de fluxo de controle
    - ↳ diferentes "caminhos" do código representados graficamente
  - ↳ Fluxo de Controle, Fluxo de Dados e Complexidade //
  - ↳ Análise de Cobertura de Critérios //

- TÉCNICA BASEADA EM DEFEITOS
  - ↳ Injeção proposital de defeitos
  - ↳ Teste de Mutação
    - ↳ Geração, a partir do programa principal, um conjunto de "programas mutantes" (com defeitos) e em seguida, define um conjunto de casos de teste que "matem" esses programas (corrijam) //

geralmente usado em sistemas críticos!

* Automação de Teste de Soft.

↳ Automatizar o processo de teste, além de ser mais prático, garante confiabilidade às versões anteriores do software, redução de custos e adequação aos critérios e técnicas.

↳ O que automatizar?

- SELEÇÃO (ALEATÓRIA OU NÃO) DE ENTRADAS
- EXECUÇÃO DO PROGRAMA SOBRE A ENTRADA DE TESTE
- ESPECIFICAÇÃO DO ORÁCULO DE TESTE * ⟶ + complexa

* Principais Ferramentas

- JUNIT
- APACHE JMETER
- MOCKITO
- SE

* Boas Práticas de automação

→ AÇÃO SUT POR TESTE - concentre-se em uma única habilidade do SUT por teste

SUT = System under testing

→ ARRANGE, ACT, ASSERT - sempre divida o teste em 3 partes (TRIPLE A)

→ DIVIDA OS TESTES POR COMPORTAMENTO - não assegure comportamentos distintos em um único teste

→ RECORRA A GERADORES - use geradores de teste para destacar apenas o que importa no teste (de forma inteligente!)

OBS: Embora os testes não garantam a ausência de defeitos, desempenham um papel essencial para elevar a qualidade do produto //