Relatório de Estratégia de Testes

Tecnologia e Ferramentas Recomendadas

Olá! Compartilho abaixo as escolhas de ferramentas e justificativas para estruturar a estratégia de testes do projeto, considerando seus desafios técnicos e o perfil dos times. O objetivo é garantir qualidade, produtividade e facilidade de manutenção, sem sobrecarregar ninguém no processo.

1. Contexto do Projeto

Temos uma plataforma web robusta, composta por frontend em React, backend híbrido em .Net e Java, além de várias integrações com APIs externas, o que traz complexidade, especialmente nos testes de integração e contratos. Os times são enxutos, com um ou dois QAs por equipe, então a automação precisa jogar a nosso favor.

2. Decisões para o Frontend (React e React Native)

- Jest + React Testing Library

São as ferramentas ideais para o dia a dia dos desenvolvedores frontend. O Jest é rápido, fácil de configurar e já faz parte do ecossistema React. A React Testing Library traz uma abordagem mais saudável, onde testamos a aplicação como nosso usuário realmente a utiliza, e não apenas sua implementação interna. Isso nos ajuda a evitar quebra de testes por mudanças banais no código.

- Cypress (Web) e Detox (React Native)

Já para simular nossos usuários navegando de ponta a ponta, Cypress brilha no frontend web e Detox faz esse mesmo papel com ótimos resultados no React Native. Ambos conseguem automatizar testes funcionais com excelente integração aos pipelines CI, fornecendo vídeos, logs e prints que ajudam muito na análise dos erros.

3. Testes Indicados para o Backend (.Net e Java)

- JUnit (Java) & NUnit/xUnit (.Net)

Podem ser integrados facilmente nos fluxos de desenvolvimento. O mesmo vale para Mockito (Java) e Moq (.Net), que vão nos permitir isolar dependências usando mocks nos unitários, agilizando feedback e focando sempre na funcionalidade correta.

- Testcontainers

Com APIs de terceiros envolvidas, precisamos simular integrações e bancos em ambiente controlado. O Testcontainers permite criar containers descartáveis para serviços e APIs mockadas durante os testes, elimina aquele clássico “na minha máquina funciona”.

4. Integração e Contratos entre Serviços

- Postman / Newman

Doces parceiros na validação rápida de APIs e fluxos de integração. Os QAs e Devs podem montar coleções reutilizáveis, versionar junto com o código e rodar nos pipelines para feedback imediato depois de cada commit relevante.

- Pact

Aqui entra um diferencial importante. Pact é nossa "rede de segurança" para contratos entre consumidores e provedores de API. Ele garante que qualquer mudança no backend (ou nos serviços externos) não vai quebrar os consumidores sem que isso seja detectado cedo.

5. Automação & Qualidade Contínua

- GitHub Actions, Azure DevOps ou Jenkins

Nossa ideia é que todo o ciclo de testes rode automaticamente a cada push ou merge, identificando regressões rapidamente sem depender da execução manual dos QAs. A escolha da ferramenta pode variar pelo contexto da empresa, todas são robustas e de fácil adoção.

- SonarQube

Vai ajudar a monitorar padrões de código, vulnerabilidades e a cobertura dos nossos testes. Isso contribui diretamente para nosso objetivo de evitar dívidas técnicas.

6. Relatórios e Acompanhamento

- Allure Report

Muitas vezes rodamos testes automatizados, mas os resultados acabam nos logs do pipeline e ninguém vê. Allure centraliza tudo, gera dashboards visuais e deixa nosso dia a dia muito mais claro, além de facilitar a vida do QA, PO e devs de todas as disciplinas.

7. Resumo das Escolhas

Apostei em ferramentas modernas, que priorizam feedback rápido, integração fácil e baixo custo de manutenção. Com esse conjunto, conseguimos cobrir desde o teste unitário dos nossos componentes até a verificação de integrações e contratos com sistemas externos. Também garanto que qualquer integrante dos times dev, QA, PO ou stakeholders, terá acesso simples ao status e à saúde do produto.

Tudo isso pensando não só na robustez técnica, mas também em não sobrecarregar os times, permitindo que cada um se concentre no que faz de melhor.