QUALIDADE DE SOFTWARE

SOFTWARE BOM O SUFICIENTE

- Imagine que você precisa de uma bicicleta para ir ao mercado.
- Você não precisa de uma bicicleta de competição com 20 marchas – uma simples, que funcione e não quebre, já é "boa suficiente". No software, é a mesma ideia:
- 1. Faz o básico bem: Resolve o problema principal sem enfeites.
- 2. Sem gastar além do necessário: Não perde tempo com coisas que o usuário não vai usar.
- 3. Pode melhorar depois: Se precisar, dá para adicionar mais funcionalidades no futuro. Segundo os especialistas, um software assim é inteligente quando há pressa ou recursos limitados. O importante é não falhar no essencial!

CUSTO DA QUALIDADE

- Qualidade e sua ausência geram custos, sendo essencial priorizá-los. Esses custos incluem investimentos para garantir qualidade e impactos da sua falta. Para gerenciá-los, a organização deve coletar métricas, reduzir gastos e estabelecer padrões. Eles se dividem em:
- Prevenção: Gastos com gestão, planejamento, controle de qualidade, desenvolvimento de requisitos, testes e treinamento para evitar falhas.
- Avaliação: Custos de inspeção e verificação para garantir que os requisitos do projeto sejam atendidos.
- Falhas:
- Internas: Erros detectados antes da entrega, envolvendo custos de correção e reavaliação.
- Externas: Problemas após a entrega, gerando custos com retrabalho, suporte, reputação e mão de obra.

RISCOS

- Dilema da Qualidade de Software Equilibrar funcionalidade e eficiência com a mitigação de riscos é essencial para garantir segurança e integridade.
- 1. Vulnerabilidades de Segurança – Falhas podem ser exploradas por malware, como estouro de buffer, permitindo ataques ao sistema.
- 2. Privilégios Excessivos – Usuários ou apps com mais permissões que o necessário podem facilitar invasões.
- 3. Homogeneidade de Sistemas – Ambientes com um único SO são mais vulneráveis a ataques em massa.
- 4. Falta de Testes – Testes inadequados podem deixar falhas ocultas, afetando confiabilidade e desempenho.
- 5. Gerenciamento de Riscos Ineficiente – A ausência de controle adequado pode gerar atrasos e custos extras.
- 6. Ausência de Modelagem de Ameaças – Não identificar ameaças durante o desenvolvimento pode comprometer a segurança do software.

RESPONSABILIDADE E NEGLIGÊNCIA

- O que é a responsabilidade por software?

É a responsabilidade legal das empresas que fabricam softwares em questões relacionadas ao seu produto. A empresa deve se responsabilizar por danos causados por seu software, o que geralmente depende dos termos específicos descritos nos contratos, das leis da região e de como o software é utilizado.

Os 6 tipos de passivos de software

1. Negligência nos padrões de qualidade: quando o software não atende ao que foi prometido.

2. Danos por software defeituoso: quando o software contém falhas que afetam sistemas físicos ou a segurança.

3. Incumprimento de normas regulatórias: quando não cumpre normas, podendo resultar em penalidades legais.

4. Violação de propriedade intelectual: quando há uso indevido de códigos ou ideias protegidas.

5. Violação de contrato: quando o suporte prometido não é fornecido.

6. Deturpação ou fraude: quando o software é promovido com falsas promessas ou omite falhas. Garantir a qualidade e a conformidade do software é essencial para evitar riscos legais e prejuízos. Empresas devem adotar boas práticas de desenvolvimento, segurança e suporte para reduzir passivos e proteger clientes e negócios

SEGURANÇA DE SOFTWARE

- A segurança do software visa protegê-lo contra ataques maliciosos que podem comprometer sua integridade e confidencialidade. Para que seja aplicada de forma eficaz, é necessário considerar tanto aspectos técnicos quanto humanos, adotando boas práticas no planejamento do software. Alguns exemplos incluem:
- Criptografia: Utilizar métodos como AES ou RSA para cifrar dados sensíveis que transitam ou são armazenados pelo software.
- Autenticação e Autorização: Implementar protocolos como OAuth ou JWT para verificar a identidade e os privilégios dos usuários que acessam o sistema.
- Validação e Sanitização de Dados: Aplicar expressões regulares ou filtros HTML para evitar ataques de injeção de código ou cross-site scripting (XSS).
- Detecção e Prevenção de Intrusões: Utilizar ferramentas como firewalls ou IDS/IPS para bloquear ou alertar sobre tentativas de acesso não autorizado.
- Seguir boas práticas em todas as etapas do desenvolvimento é essencial para garantir a qualidade, a eficiência e a segurança do software

SEGURANÇA DE SOFTWARE

- Gerenciamento de Acessos e Segurança: Controle de permissões evita acessos não autorizados.
- Políticas de segurança (como autenticação multifator) protegem dados sensíveis.
- Manutenção e Atualizações: Atualizações regulares corrigem vulnerabilidades e melhoram a performance.
- Backups garantem recuperação de dados em caso de falhas.
- Otimização de Recursos: Monitoramento de servidores e bancos de dados melhora a eficiência.
- Ajuste de configurações: evita desperdício de recursos.
- Conformidade e Auditoria: Registro de logs permite rastrear alterações e identificar falhas.
- Conformidade com normas (LGPD, GDPR) evita penalidades legais.Suporte e AtendimentoResolução rápida de incidentes minimiza impactos nos usuários.Implementação de melhorias contínuas mantém a satisfação do usuário.

OBRIGADO

HENRYK BAGDANOVICIUS ROZA

OLIVIA FRANKIW DE CARVALHO

GEOVANE AUGUSTO

JOÃO LUÍS SANTANA BOREAN

NATAN FERNANDES ARAUJO IBIAPINA