**3)O que é débito técnico e quais ferramentas podem ser usadas para verificar débito técnico?**

Débito técnico são ações tomadas para agilizar a entrega de uma funcionalidade ou um projeto; O débito técnico ocorre em qualquer situação que uma solução ou funcionalidade seja entregue sem ser totalmente finalizada, o que geralmente acontece para atender pressão dos clientes por prazos, existem 4 formas de débito técnico;

•          **Imprudente intencional:** o time de desenvolvimento sabe da existência de [erros](https://cloudster.com.br/blog/maiores-erros-de-nuvem-que-voce-nao-pode-cometer/?utm_source=blog-72049&utm_medium=referral) no código, porém lança a solução sem resolver os problemas existentes;

•          **Imprudente não intencional:** programadores usam linguagens de programação que não conhecem bem, gerando necessidade de correções futuras;

•          **Consciente intencional:** uma implementação é feita ou solução enviada com problemas, mas dentro do prazo. Nesse caso os desenvolvedores se comprometem a corrigir erros no futuro;

•          **Consciente não intencional:** o time conclui um projeto, mas apenas percebe as falhas no código posteriormente.

**5) Quando usar o controle de versão centralizado e quando usar o distribuído? Pontos positivos e negativos de cada um.**

**Quando usar o Controle de Versão Centralizado: Ponto positivo.**

**1**. **Projetos Pequenos e Simples**: Para projetos pequenos, onde a equipe é compacta e complexidade não é alta, um sistema centralizado pode ser suficiente.

**2**. **Equipes Inexperientes:** Se a equipe não estiver familiarizada com sistemas de controle de versão mais complexos, um sistema centralizado pode ser mais fácil de aprender e usar inicialmente.

**Quando não usar o Controle de Versão Centralizado: Ponto negativo.**

**1.Fluxo de Trabalho Restrito:** Projetos maiores com fluxos de trabalho complexos que envolvem múltiplas ramificações (branches) e recursos simultâneos podem enfrentar restrições ao usar um sistema centralizado. O processo de mesclagem e colaboração pode ser mais complicado e menos flexível.

**2.Conflitos Complexos**: Quando várias pessoas estão trabalhando simultaneamente em partes diferentes do código e tentam fazer check-in de suas alterações no repositório central, podem surgir conflitos difíceis de resolver. Isso pode levar a atrasos e confusão na equipe.

**Quando usar o Controle de Versão Distribuído: Ponto positivo**

**1. Projetos Colaborativos e Complexos**: Projetos que envolvem várias equipes ou colaboradores externos se beneficiam do controle de versão distribuída, pois cada um pode trabalhar em sua cópia local e depois sincronizar as alterações.

**2.Recuperação de Dados**: Cada desenvolvedor possui uma cópia completa do histórico, tornando mais fácil recuperar versões anteriores ou desfazer alterações.

**Quando usar o Controle de Versão Distribuído: Ponto negativo**

**1.Complexidade Inicial**: O uso de um sistema distribuído pode ser mais complicado de entender e configurar, especialmente para equipes novas ou menos experientes.

**2.Aprendizado Adicional:** Os membros da equipe podem precisar aprender novos conceitos e comandos específicos do sistema de controle de versão distribuída, o que pode exigir tempo.