






Notas de Introdução ao Clean Code

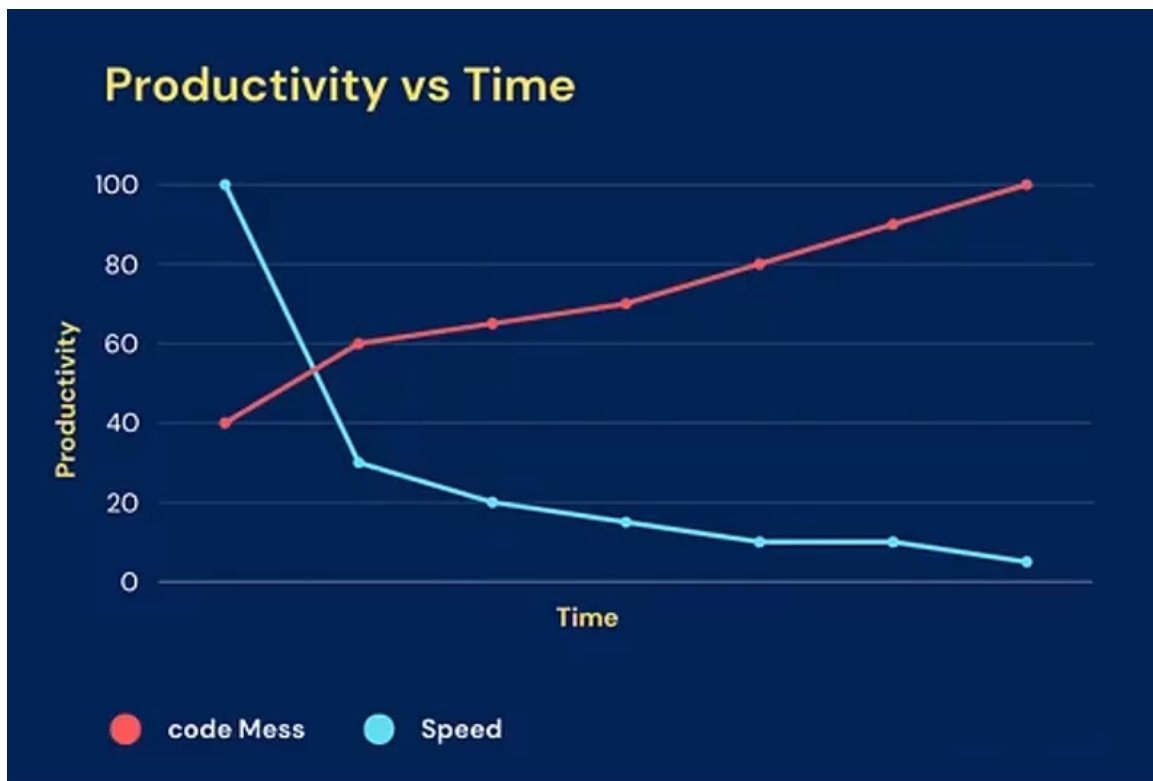
Type	Info
Topics	 <u>Studies</u>
Notebooks	 <u>Programação Profissional com Clean Code</u>
Related Notes	 <u>Ementa do Curso</u>

O que é Clean Code?

- Conjunto de princípios e práticas; fácil de entender e modificar; promove simplicidade, clareza e eficiência; código é pra ser uma documentação

Custo de ter um código ruim

- Impacto na produtividade: cada alteração no código se torna arriscada
- Aumento dos custos: necessidade de corrigir bugs; lidar com problemas de desempenho
- Ciclo de dependência: redesign do sistema devido ao ciclo de desenvolvimento caro e ineficaz



Impactos negativos de reescrever o código

- Tempo e recursos: grandes mudanças introduzem muitos riscos
- Risco de falhas: novo sistema pode ter novos bugs ou nunca ter sido concluído
- Moral da equipe

Estratégias alternativas

- Refatoração contínua: melhoria incremental através de uma mudança por contexto
- Divisão de responsabilidades: quebrar a aplicação; dividir por negócio e funcionalidade
- Testes automatizados: novas mudanças não introduzam bugs

Atitudes

- Importância: mentalidade do desenvolvedor impacta na qualidade do código; necessário responsabilidade e ética profissional
- Responsabilidade

- Qualidade: desenvolvedor deve se sentir responsável pela clareza, manutenção e eficiência do código
- Comunicação: informar gestores e stakeholders sobre implicações de decisões técnicas
- Atitude proativa
 - Refatoração contínua: não esperar que o código fique insustentável para melhorá-lo
 - Feedback: buscar feedback constante sobre o código; estar aberto a aprender as melhores práticas
- Boas e más atitudes
 - Boa atitude: desenvolvedor que se preocupa com a legibilidade fazem revisões e incentiva boas práticas
 - Má atitude: código sem pensar na manutenção futura; deixa dívidas técnicas acumularem

Dilema da produtividade

- Produtividade x qualidade: é mais rápido fazer certo na primeira vez
- Impacto do código sujo
 - Débito técnico
 - Manutenção e correção de bugs
- Estratégias para equilibrar velocidade e qualidade
 - Test-Driven Development (TDD): garantir que novas modificações tenham cobertura de código
 - Revisões de código
 - Pair programming
 - Regra do escoteiro: deixar o acampamento mais limpo; refatorar o que já foi escrito