



SOFTEX
PERNAMBUCO

 **Softex**

MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO



Engenharia de software: principais conceitos!

A engenharia de software é um amplo campo de conhecimento ligado ao gerenciamento de projetos, por isso não esgotaremos seu conteúdo neste curso, o objetivo principal é aprendermos sobre seus conceitos e principais diretrizes.



1. **O Que é Engenharia de Software?** - A base de tudo.
2. **Ciclo de Vida do Software (SDLC)** - Como o software é construído.
3. **Ferramentas e Práticas Essenciais** - O dia a dia do desenvolvedor.
4. **Escrevendo Código de Qualidade** - A arte do código limpo.
5. **Qualidade e Testes de Software** - Garantindo que tudo funcione.
6. **Próximos Passos** - Para onde ir a partir daqui.



O que é Engenharia de Software?

- Programação != Engenharia de software!
 - Programar é o ato de escrever instruções para computadores.
- Engenharia de software é a aplicação de princípios de engenharia e gerenciamento de projetos, é como construímos software com qualidade e durabilidade.
- A engenharia de software aplica princípios como:
 - Projetar
 - Desenvolver
 - Testar
 - Manter
 - Evoluir





Ciclo de vida do Software

- O Ciclo de Vida de Desenvolvimento de Software (SDLC) é um processo estruturado para produzir software de alta qualidade.
- Modelos Comuns:
 - Cascata (Waterfall): **(Muito comum no passado)**
 - Linear e sequencial.
 - Requisitos → Projeto → Implementação → Teste → Implantação.
 - Rígido e com pouca flexibilidade para mudanças.
 - Ágil (Agile): **(modelo mais comum atualmente)**
 - Iterativo e incremental.
 - Foco em entregas contínuas de valor, colaboração e adaptação a mudanças.





Ágil com Scrum

Scrum é o framework ágil mais popular. Ele organiza o trabalho em ciclos chamados Sprints (geralmente de 2-4 semanas).

- Papéis Principais:
 - **Product Owner (PO):** Define "o que" será construído.
 - **Scrum Master:** Garante que o processo Scrum seja seguido. Remove impedimentos.
 - **Time de Desenvolvimento:** Constrói o produto.
- Cerimônias (Eventos):
 - **Planning:** Planejamento do que será feito na Sprint.
 - **Daily Scrum:** Reunião diária rápida de alinhamento.
 - **Sprint Review:** Demonstração do que foi construído na Sprint.
 - **Retrospective:** Discussão sobre como melhorar o processo.

O objetivo geral desta metodologia é o feedback rápido e adaptabilidade a mudanças!





Git

Nosso já conhecido amigo, vamos apenas lembrar sua importância neste contexto:

- **Trabalho em Equipe:** Permite que vários desenvolvedores trabalhem no mesmo projeto sem sobrescrever o trabalho uns dos outros.
- **Histórico Completo:** Você sempre sabe quem mudou o quê, quando e por quê.
- **Segurança:** É como um "backup" infinito do seu projeto. Você pode experimentar sem medo de quebrar o que já funciona.





Clean Code

O que é "Código Limpo"?

- **Legível e Entendível:** Fácil de seguir a lógica.
- **Manutenível:** Fácil de corrigir, alterar e ampliar.
- **Nomes Significativos:** Variáveis e funções que descrevem seu propósito.
- **Funções Pequenas e Focadas:** Cada função faz apenas UMA coisa.

É importante escrever código que seja legível não só para os computadores, mas também por outros programadores. Recomendo fortemente o livro: “Clean Code: A Handbook of Agile Software Craftsmanship” de Robert C. Martin, Michael C. Feathers, Timothy R. Ottinger.





O essencial para código!

- **KISS (Keep It Simple, Stupid)**
 - "Mantenha as coisas simples, estúpido".
 - Prefira a solução mais simples que funcione. Evite complexidade desnecessária.
- **DRY (Don't Repeat Yourself)**
 - "Não se repita".
 - Evite duplicação de código. Se você copia e cola um trecho de código, provavelmente deveria transformá-lo em uma função ou classe reutilizável.
- **YAGNI (You Ain't Gonna Need It)**
 - "Você não vai precisar disso".
 - Não adicione funcionalidades pensando "talvez um dia a gente precise". Implemente apenas o que é necessário agora.





Testando!



Por que testar?

- Para garantir que o código faz o que deveria fazer.
- Para encontrar bugs antes que o usuário encontre.
- Para dar confiança ao fazer alterações (refatoração). Testes são sua rede de segurança!



Testando!

Existem 3 tipos de testes principais na pirâmide de testes.

- **Testes de Unidade (Base):**
 - Testam a menor parte do código (uma função, um método).
 - Rápidos e baratos. Devemos ter muitos.
 - Quanto mais se investe na base, menos dor de cabeça você deve ter com o topo!
- **Testes de Integração (Meio):**
 - Testam como diferentes partes do sistema interagem.
- **Testes End-to-End (E2E) (Topo):**
 - Simulam a jornada completa do usuário na aplicação.
 - Lentos e caros. Devemos ter poucos, mas importantes.

Obs: testes automatizados não excluem a necessidade de testes manuais!





Próximos passos

Engenharia de software é um tópico muito amplo para tratar de forma completa num curso tão curto por isso:

- Recomendo a pesquisa das referências contidas no próximo slide, a leitura dos livros e o acesso aos sites referência no assunto com certeza serão muito frutíferos!
- Nós veremos padrões de projetos, entre outros assuntos, que estão ligados diretamente ao que será realizado no curso, por isso não é preciso ansiedade de conhecer tudo sobre o tema
- Esta é uma área muito ampla e que muitas pessoas se especializam, estude e veja se também é do seu interesse.





Referências



Livros

- **"Engenharia de Software"** - Ian Sommerville (A referência absoluta do tema!)
- **"Engenharia de Software: Uma Abordagem Profissional"** - Roger S. Pressman, Bruce R. Maxim
- **"O Mítico Homem-mês: Ensaios Sobre Engenharia de Software"** - Frederick P. Brooks Jr. (Ótimo contexto histórico)
- **"O Programador Pragmático"** (The Pragmatic Programmer) - Andrew Hunt e David Thomas
- **"Código Limpo"** (Clean Code) - Robert C. Martin, Michael C. Feathers, Timothy R. Ottinger.
- **"Pro Git"** - Scott Chacon e Ben Straub (disponível online)
- **"Succeeding with Agile"** - Mike Cohn (popularizou o conceito da Pirâmide de Testes)
- **"Working Effectively with Legacy Code"** - Michael Feathers (interessante, mas o menos importante da lista)



Referências

Conteúdo online:

- **Atlassian Agile Coach:** atlassian.com/br/agile (para conceitos de Ágil e Scrum)
- **O Guia do Scrum:** scrumguides.org (a fonte oficial do Scrum)
- **Site Oficial do Git:** git-scm.com
- **Learn Git Branching:** learngitbranching.js.org (tutorial interativo de Git)
- **Blog do Martin Fowler:** martinfowler.com (artigos sobre arquitetura e design de software)
- **Roadmap.sh:** roadmap.sh (roteiros de estudo para desenvolvedores)
- **Refactoring.Guru:** refactoring.guru (explicações visuais sobre padrões de projeto)
- **Definição do IEEE:** Padrão oficial para a definição de Engenharia de Software.

