Visão geral da engenharia de software

Software está presente em quase tudo

Operações bancárias, comunicação, comércio eletrônico, serviços públicos, gestão de empresas, equipamentos médicos, sistemas embarcados em dispositivos — praticamente toda atividade contemporânea depende de software funcionando de maneira confiável.

Desenvolver esses sistemas profissionalmente exige muito mais do que escrever código.


FUNCIONALIDADES

Diagnóstico de um projeto de software

O caso: Sistema de Biblioteca

Uma biblioteca de Salvador contratou uma pequena equipe para desenvolver um sistema web utilizando React, Node.js e PostgreSQL.

Após uma conversa inicial com o cliente, a equipe começou a implementar o cadastro de livros e usuários, definiu empréstimos com prazo fixo de sete dias e limitou cada usuário a três livros.

No primeiro deploy para testes, o cliente informou que professores tinham prazo diferente, algumas obras não poderiam ser emprestadas, era necessária uma fila de reserva e atrasos deveriam bloquear novos empréstimos. Parte do sistema precisaria ser refeita.

A Tarefa: Analise e Identifique Problemas

Analisem o caso e identifiquem três problemas chave.

1. Qual é o problema?

2. Que consequência ele pode provocar?

3. O que deveria ter sido feito de maneira diferente?

O que deveria ter sido diferente?

Ao analisar o caso do sistema de biblioteca, diversas oportunidades de melhoria surgem, principalmente na fase inicial de levantamento de requisitos e validação.

Problema identificado	Possível consequência	O que deveria ter sido feito?
A equipe começou a implementar após uma compreensão inicial insuficiente.	Regras importantes do negócio ficaram de fora e foram ignoradas.	Levantar e validar melhor as necessidades e requisitos completos com o cliente **antes** da implementação.
Regras como prazos diferentes, restrições de empréstimo e bloqueios por atraso não foram identificadas.	Funcionalidades foram construídas com regras incorretas, gerando retrabalho.	Identificar, documentar e registrar todas as regras de negócio do sistema de forma clara e acessível.
A fila de reserva não foi prevista no escopo inicial.	Uma necessidade importante do usuário final ficou ausente, impactando a satisfação.	Confirmar todas as funcionalidades esperadas e prioridades com o cliente através de protótipos ou descrições.
As inconsistências só apareceram no primeiro deploy para testes, já em estágio avançado.	Parte significativa do sistema precisou ser refeita, causando atrasos e custos adicionais.	Validar decisões e resultados incrementalmente ao longo do desenvolvimento com feedback contínuo do cliente.

Mesmo que todos os programadores sejam tecnicamente muito bons, esses problemas ainda poderiam acontecer? Por quê?

Engenharia de Software é mais do que programação

Engenharia de Software organiza conhecimentos, métodos, ferramentas e práticas para desenvolver software de forma disciplinada, sistemática e mensurável.

Ela engloba atividades como:

- Compreensão das necessidades
- Especificação e análise
- Projeto e implementação
- Verificação e validação
- Gerenciamento, manutenção e evolução

Programar é uma atividade essencial, mas não representa todo o desenvolvimento de software.

Por que desenvolver software exige organização?

Sistemas maiores e mais utilizados envolvem mais funcionalidades, dados, integrações e regras. Quanto maior a complexidade, maior a necessidade de organização. A Engenharia de Software torna o desenvolvimento mais controlável e capaz de produzir software com **qualidade, confiabilidade e eficiência**.

Software pode ser produto e serviço

Software como produto

- Disponibiliza capacidades computacionais ao usuário
- Recebe, transforma, armazena, apresenta ou transmite informações
- Precisa atender necessidades definidas

Exemplo: aplicativo instalado no dispositivo do usuário

Software disponibilizado como serviço

- Permanece acessível ao usuário por meio da rede
- Depende de operação, disponibilidade, atualização e suporte contínuos
- Continua exigindo requisitos, projeto, código, testes, documentação e evolução

Exemplo: sistema acessado pelo navegador ou aplicativo conectado

O software é um sistema lógico: ele é desenvolvido, não manufaturado, e não sofre desgaste físico pelo uso.

Da necessidade à evolução

1

Necessidade e requisitos

Compreender problema, usuários, funções e restrições

2

Análise e projeto

Organizar, representar e estruturar a solução

3

Implementação

Transformar decisões em componentes executáveis

4

Verificação e validação

Testar comportamentos e confirmar que atende às necessidades

5

Implantação e evolução

Disponibilizar, corrigir, adaptar e incorporar mudanças

Esta é uma visão conceitual. As atividades podem ser organizadas e revisitadas de formas diferentes conforme o processo utilizado.

Desenvolver software é trabalho coletivo

Diferentes participantes contribuem com conhecimentos distintos:

- **Usuários e gestores** conhecem as necessidades do contexto
- **Analistas** estruturam requisitos e modelos
- **Desenvolvedores** implementam a solução
- **Profissionais de teste** verificam comportamentos
- **Infraestrutura e operação** conhecem o ambiente de funcionamento

Uma mesma pessoa pode assumir mais de uma responsabilidade em equipes pequenas.

O desenvolvimento profissional é uma atividade técnica e também uma atividade de comunicação.

Organizar o trabalho reduz a improvisação

Planejamento

Organiza objetivos, prioridades,
entregas e responsabilidades da equipe.

Requisitos

Registram necessidades, funções e
restrições do sistema antes de implementar.

Documentação e modelos

Preservam informações e decisões necessárias
para compreender e continuar o trabalho.

Testes e validação

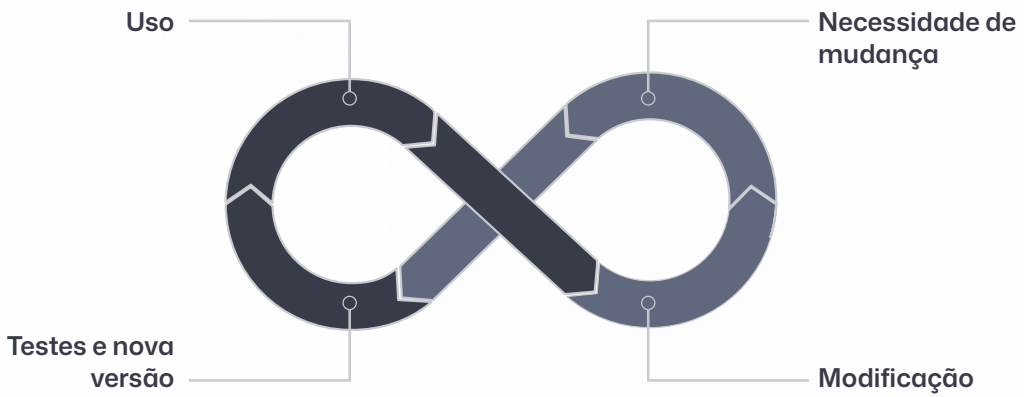
Permitem confrontar o sistema construído
com aquilo que deveria realizar.

Documentar não significa produzir registros sem finalidade: significa preservar informações necessárias ao projeto.

A entrega não encerra a vida do software

O software não sofre desgaste físico pelo uso, mas o **contexto em que ele funciona muda**. Um sistema pode precisar de alterações para:

- Corrigir falhas identificadas após a entrega
- Responder a novas necessidades dos usuários
- Adaptar-se a mudanças tecnológicas ou de integração
- Continuar útil e relevante para seus usuários



Uma solução precisa funcionar hoje e também ser compreensível o suficiente para ser modificada amanhã.

Relembrado o caso da biblioteca

A equipe começou a programar sem discutir necessidades, manteve registro incorreto de funcionalidades e planejou testar o sistema somente no final.

Problema	Princípio relacionado	Consequência provável
Começar sem compreender necessidades	Ausência de levantamento e validação de requisitos	Risco de construir funções diferentes das realmente necessárias
Decisões mantidas apenas na memória	Ausência de documentação adequada	Dificuldade para comunicar, revisar e continuar o trabalho
Testar somente ao final	Verificação tardia	Problemas percebidos apenas quando grande parte da solução já foi construída
Sem responsabilidades e organização claras	Ausência de planejamento e coordenação	Dificuldade para acompanhar decisões, prioridades e entregas

O que levar desta aula

1 Engenharia de Software é maior que programação

Envolve análise, projeto, verificação, gerenciamento, documentação e evolução.

2 Software é um sistema lógico

Pode ser produto ou serviço. Não sofre desgaste físico, mas evolui continuamente.

3 O desenvolvimento começa pela compreensão

Entender necessidades antes de implementar é fundamental, não opcional.

4 Complexidade exige colaboração organizada

Planejamento, comunicação, documentação e verificação reduzem a improvisação.

5 Software é mantido e evoluído após a entrega

A primeira versão raramente é a última. Clareza e documentação tornam a evolução viável.

Por que começar a programar não significa necessariamente começar corretamente um projeto de software?

Obrigado e até a próxima!

Uma ótima volta para casa!