

Ciclos de vida de *software*

Disciplina: Qualidade de *Software*



Conteúdos:

Ciclos de vida de *software*.

Habilidade(s):

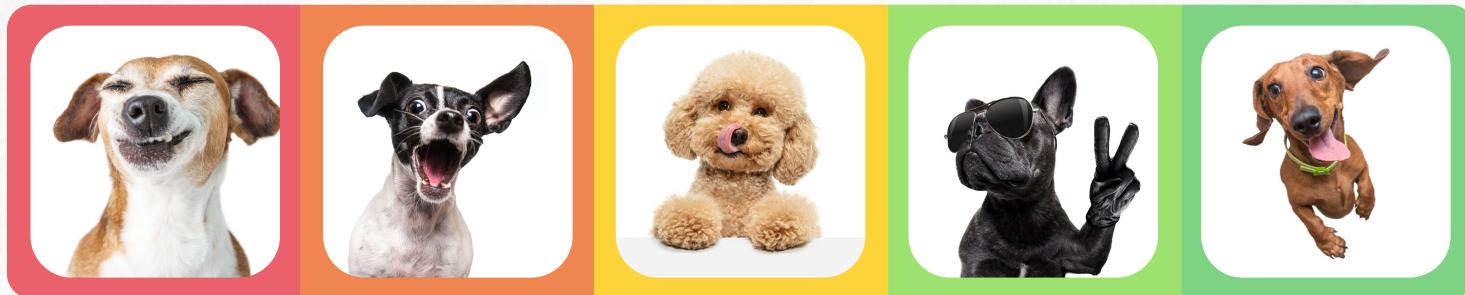
- Compreender os métodos de ciclo de um desenvolvimento de *software*;
- Entender a correspondência entre projeto solicitado e projeto entregue.

Bloco 1

Vamos compreender os aspectos principais sobre os ciclos de vida de um *software*!

Quem é você?

Em uma escala de **doguinho**, quem é você quando precisa se planejar para alcançar uma meta?

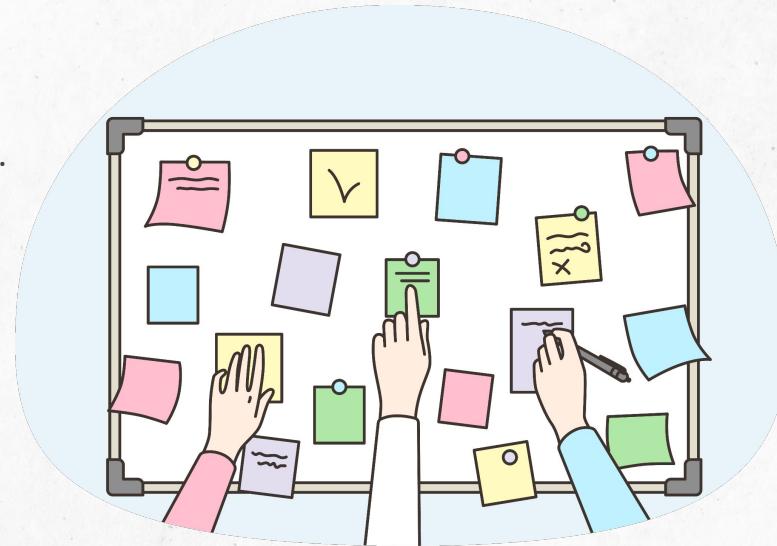


| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|----|

A importância do processo

Na nossa vida, é essencial seguir etapas específicas para alcançar a eficácia e o sucesso de um projeto.

Durante o desenvolvimento de um *software*, não é diferente. É para isso que existem os **ciclos de vida de um software**.



Ciclos de vida de um *software*

Referem-se aos diferentes estágios pelos quais um *software* é planejado, desenvolvido, testado, implantado, e, eventualmente, aposentado.

Esses ciclos visam **organizar e estruturar o desenvolvimento, a manutenção e o gerenciamento do software ao longo do tempo.**



O que compõe os ciclos da vida de um *software*?

Embora possam existir vários processos, as principais etapas dos ciclos de vida são:

1

Especificação: comprehende as necessidades e os requisitos do cliente ou usuário.

2

Projeto e implementação: com base nos requisitos definidos, os *designers* e desenvolvedores criam o projeto detalhado do *software*.

3

Validação: o *software* precisa ser validado para garantir que atenda aos requisitos.

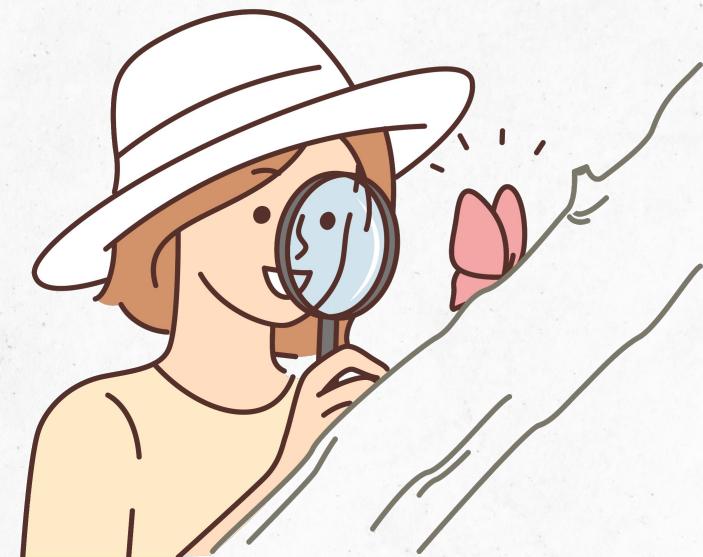
4

Evolução: lida com a adaptação contínua do *software* para atender a essas mudanças.

Vamos acompanhar a vida de Bitfly

Assim como os ciclos de um *software*, a borboleta passa por várias fases ao longo de sua vida.

Para entender melhor como funcionam todas as etapas do desenvolvimento de um programa, iremos acompanhar a história de **Bitfly**.

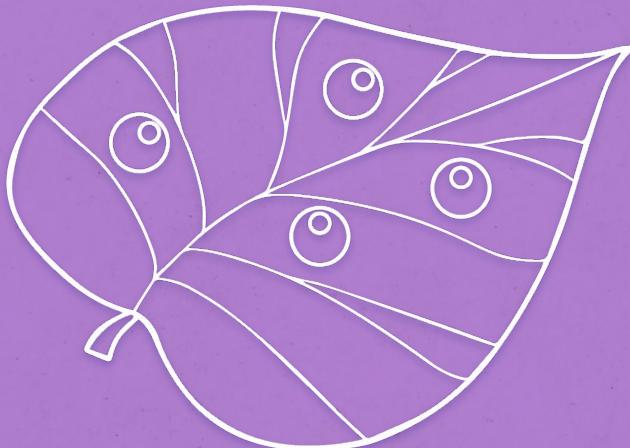


Bloco 2

Etapa de especificação de um *software*.

Ovo

É a primeira fase da vida da borboleta.



Vamos imaginar o seguinte cenário

Imagine que você é um arquiteto e foi contratado para construir uma casa sob medida para um cliente. Para atender a este pedido, o que você faz?



Entender para atender

Imagine que você é um arquiteto e foi contratado para construir uma casa sob medida para um cliente. Para atender a este pedido, o que você faz?

Antes de começar, você precisa entender exatamente o que o cliente deseja, quais são os requisitos para a casa e quais são as restrições da construção.



Começando pelo começo

Essa etapa corresponde à **especificação inicial** do *software* e equivale à fase do **ovo** da Bitfly, pois é a **primeira etapa** de um projeto.

Assim como o ovo contém as informações básicas do ser vivo, a especificação define o essencial do *software*, incluindo os requisitos fundamentais.



Não segui a etapa de especificação. E agora?

Por ser uma etapa inicial, erros ocasionados nesse processo podem gerar problemas futuros para o seu projeto. Veja o que pode acontecer.

Baixa qualidade

Comunicação falha

Gastos desnecessários

Caos na equipe

Aquela atrasada básica

Perda de credibilidade



Etapas para uma boa especificação

Por isso, durante essa etapa, é necessário seguir alguns passos essenciais. Se liga!



Estudo de viabilidade: você verifica se é possível realizar o projeto.



Levantamento e análise de requisitos: você se reúne com o cliente e faz várias perguntas para entender as suas necessidades e os seus desejos.



Especificação de requisitos: com base nas informações do cliente, você elabora uma lista detalhada de requisitos.



Validação de requisitos: antes de começar a construir, você mostra a lista de requisitos ao cliente para garantir que tudo esteja correto.

Vamos praticar?

Formem um grupo de quatro pessoas para realizar a atividade.

Primeiro momento

Duas pessoas devem criar uma ideia de aplicativo fictício.

Segundo momento

Com a ideia elaborada, as outras duas pessoas devem realizar um atendimento fictício à dupla do primeiro momento, para que possam desenvolver uma etapa de especificação à ideia elaborada.



Bloco 3

Etapa de projeto e implementação.

Vamos relembrar?

Como você viu, durante a etapa de especificação, é necessário seguir alguns passos para realizar o processo de forma efetiva.

Você consegue se lembrar do que é realizado em cada fase? Analise o caso abaixo e tente responder corretamente.

O "Sem panho, sem ganho" é um aplicativo cujo objetivo é ajudar os usuários a monitorar sua saúde, registrar atividades físicas, controlar a dieta e receber dicas personalizadas para melhorar o seu bem-estar geral.



E aí, acertou?

O "Sem panho, sem ganho" é um aplicativo cujo objetivo é ajudar os usuários a monitorar sua saúde, registrar atividades físicas, controlar a dieta e receber dicas personalizadas para melhorar o seu bem-estar geral.

Estudo de viabilidade: o aplicativo é viável?

Levantamento e análise de requisitos: quais são os requisitos?

Especificação de requisitos: definição clara dos requisitos.

Validação de requisitos: os requisitos atendem às necessidades?



Lagarta

Agora, a borboleta passa por uma fase de larva, também conhecida como lagarta. Durante essa etapa, a lagarta cresce e se alimenta, acumulando energia para a próxima fase.



Chegou a hora de implementar

Após a fase de formação inicial, a Bitfly precisa desenvolver-se.

A fase de **implementação** do projeto é semelhante à etapa da lagarta na vida de uma borboleta, pois é aqui que ela acumulará energia para crescer.



Quando a ideia sai da cabeça

Nessa etapa, a equipe precisa transformar os requisitos elaborados em um **programa executável**.

Desta forma, o **projeto é construído** e envolve:

- dados do sistema;
- interfaces entre os componentes do sistema;
- algumas vezes, o algoritmo utilizado.



Atividades durante a elaboração do projeto

É ao longo do projeto que a equipe irá interagir entre si para dar **forma e detalhes** ao *software*. Sendo assim, ela costuma estar envolvida em:

projeto de arquitetura

projeto de componentes

especificação abstrata

projeto de estrutura de dados

projeto de interface

projeto de algoritmos

Na hora de se organizar...

Quando nos planejamos, costumamos ter diferentes hábitos de organização. Para você, o mais importante é **ter uma lista flexível** ou **estruturar cada passo** do projeto?



Flexibilidade

VS



Estrutura

Métodos do projeto

Cada equipe costuma ter uma abordagem específica durante o projeto de um *software*. Esses métodos costumam se aplicar à realidade da equipe e não estão ligados diretamente ao sucesso de um produto.

Sendo assim, não existe um **pior** ou **melhor** método!

Flexibilidade

À medida que uma equipe avança no desenvolvimento de um *software*, é comum esquecer de anotar todas as mudanças. Isso garante uma maior flexibilidade, mas pode ser difícil de entender e replicar posteriormente.

VS

Estrutura

Esse método oferece um conjunto de diretrizes, notações e modelos que ajudam a planejar o projeto. Isso torna o desenvolvimento excessivamente formal, mas revela como as partes do sistema se relacionam e fluem entre si.

Uma alternativa são os métodos ágeis

Métodos ágeis são uma abordagem moderna e iterativa no desenvolvimento de *software*, focada em **flexibilidade, colaboração e adaptação**.

Baseia-se em ciclos curtos de desenvolvimento, chamados *iterações* ou *sprints*, com planejamento, execução, revisão e ajuste.

No *scrum*, um dos métodos ágeis mais conhecidos, uma equipe multidisciplinar é organizada em papéis específicos e dividida em pequenas tarefas.

Modelo Big Bang

Nesse método, o *software* é concebido através de uma “explosão”.

Dessa forma, **não há** planejamento, cronograma ou processo formal.

Aqui, todo o esforço é gasto com **codificação**.



Modelo Constrói e Corrige

Derivado do modelo *Big Bang*, nesse método, o *software* é desenvolvido apenas com um projeto simples.

O ideal é que seja **utilizado em protótipos** ou projetos que serão descartados posteriormente.

Constrói

Fase de testes

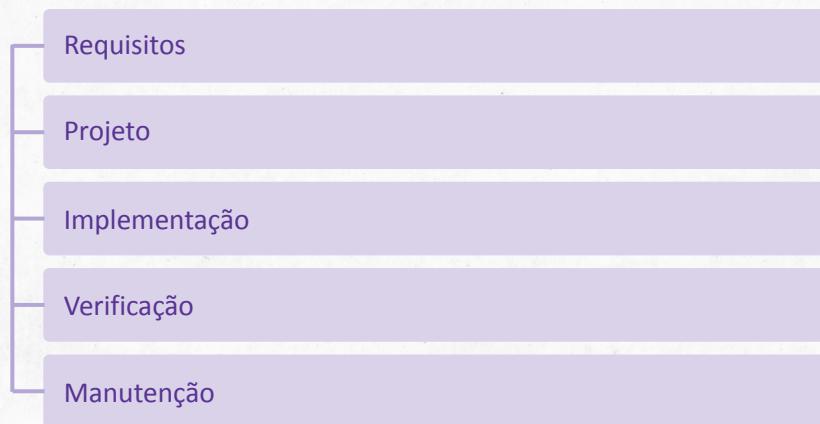
Corrige

Constrói

Modelo Cascata

É um método linear e sequencial, onde cada etapa é feita de maneira ordenada.

Aqui, o trabalho é dividido em **etapas**, que são iniciada somente após a conclusão da fase anterior.



Modelo Espiral

No início do projeto, apenas as funcionalidades importantes são estabelecidas. Dessa forma, o próximo ciclo se inicia após a finalização dos testes.

Cada ciclo espiral possui, em média, seis passos:

1 Determinar os objetivos, alternativas e restrições.

2 Identificar e resolver os riscos.

3 Avaliar as alternativas.

4 Desenvolver e testar o ciclo atual.

5 Planejar o próximo ciclo.

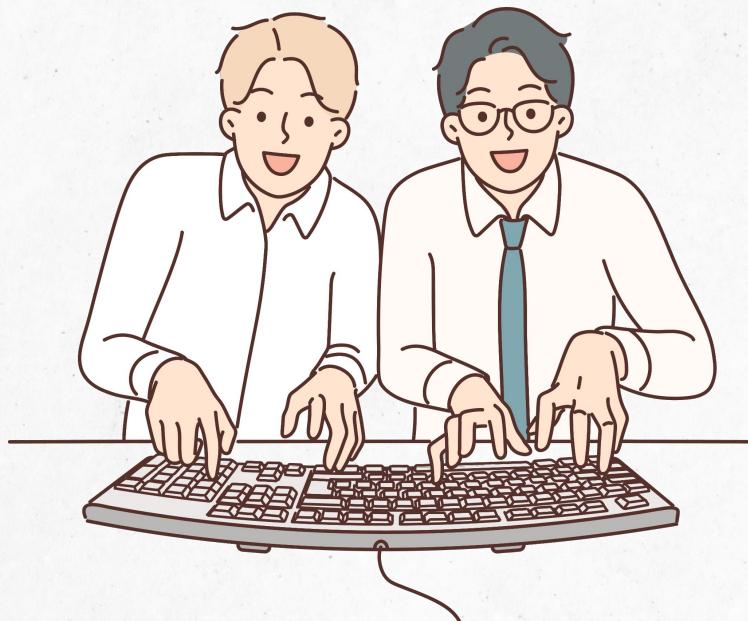
6 Definir as técnicas para o próximo ciclo.

Fase de programação e depuração

Por fim, após estabelecer o projeto do *software* e o método que será usado para desenvolvê-lo, a equipe pode realizar a sua programação.

Programar também é depurar!

Durante o projeto, os desenvolvedores também podem realizar testes nos códigos que eles mesmos produziram. Isso é chamado de **depuração**.



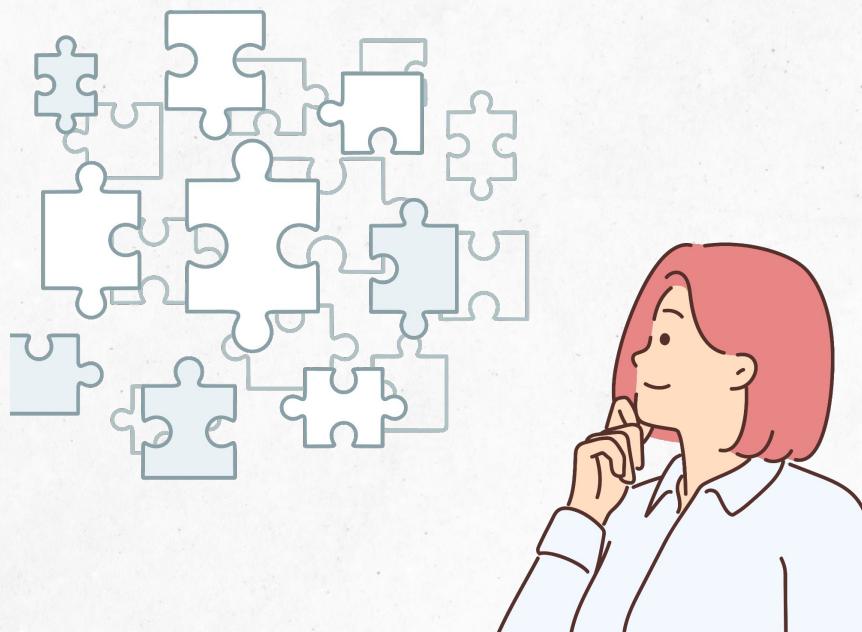
Bloco 4

Etapa de garantia de qualidade de um *software*.

Garantindo a eficácia do produto

Vamos realizar um jogo da forca? Resolva o enigma e descubra a **terceira etapa** do ciclo de vida de um *software*.

No mundo digital, ela é a garantia,
Sem ela, a segurança acabaria,
Processo de verificação, sem margem à negação,
Que palavra é essa, tão vital à ação?



Pupa

Durante essa fase, ocorrem transformações internas profundas, resultando na metamorfose da lagarta.



A importância da validação

Esta etapa assegura que o *software* criado atenda aos requisitos definidos e esteja pronto para ser entregue aos usuários finais.

Por isso, ela corresponde à **terceira fase** da vida de Bitfly: a **pupa**! Aqui, ela irá passar por mudanças internas a fim de garantir o seu desenvolvimento pleno, assim como acontece na validação.



Documentos necessários para a validação

Os documentos de teste são **essenciais** para as equipes que irão testar e validar o *software*.

A seguir, confira os mais importantes!

Plano de teste

- Costuma testar *hardwares* e *softwares* e é responsável por um fluxograma detalhado;
- É um dos oito documentos descritos na IEEE 82;
- Ele identifica, por exemplo, os itens de teste, os recursos a serem testados, as tarefas de teste, o executor de cada tarefa e o grau de independência do testador.

Casos de teste

- Costuma testar *softwares*;
- Utilizado para identificar defeitos;
- Ele deve conter, por exemplo, precondições de execução, ações e valores de entrada, resultados esperados e pós-condições de execução desenvolvidas.

Documentos necessários para a validação

Além disso, alguns outros tipos de testes podem ser realizados, como:

Reporte de incidentes (*bugs*)

Descreve os *bugs* encontrados juntamente com os passos para reproduzir.

Metas, estatísticas e resumos

Transmitem o progresso dos testes.

Bloco 5

Etapa de evolução de um *software*.

Borboleta

A evolução é constante

Assim como a borboleta adulta está pronta para interagir com o ambiente e evoluir, nesta fase, o *software* está pronto para ser utilizado!

Além disso, ele lida com a adaptação contínua às mudanças nos requisitos e no ambiente, semelhante à borboleta adulta.



Benefícios da constante evolução

Um *software* está sempre em constante evolução por várias razões que refletem as necessidades e demandas do mundo da tecnologia, dos usuários e das empresas. As suas principais vantagens são:

melhorias de desempenho

adaptações a novas tecnologias

inovações e recursos novos

segurança

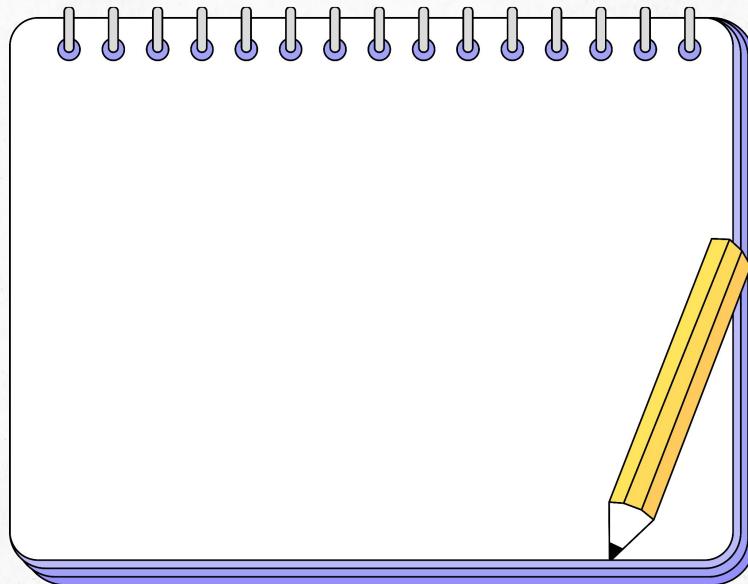
feedback dos usuários

compatibilidade



Alguns exemplos

Você consegue pensar em algum aplicativo que sofreu evoluções ao longo do tempo? Preencha junto comigo!



Bloco 6

Chegou a hora de praticar!

Baú da memória

Para o exercício dessa aula, separem-se em grupos e escolham **um aplicativo ou programa utilizado no dia a dia**. Lembrando que as equipes não podem repetir os aplicativos!

Com uma cartolina e canetas coloridas, construam uma **breve biografia** do aplicativo segundo os seus ciclos de *software*. Dessa forma, a cartolina deve conter:

Especificação: como a equipe pensou na ideia?

Implementação: como foi feito o projeto?

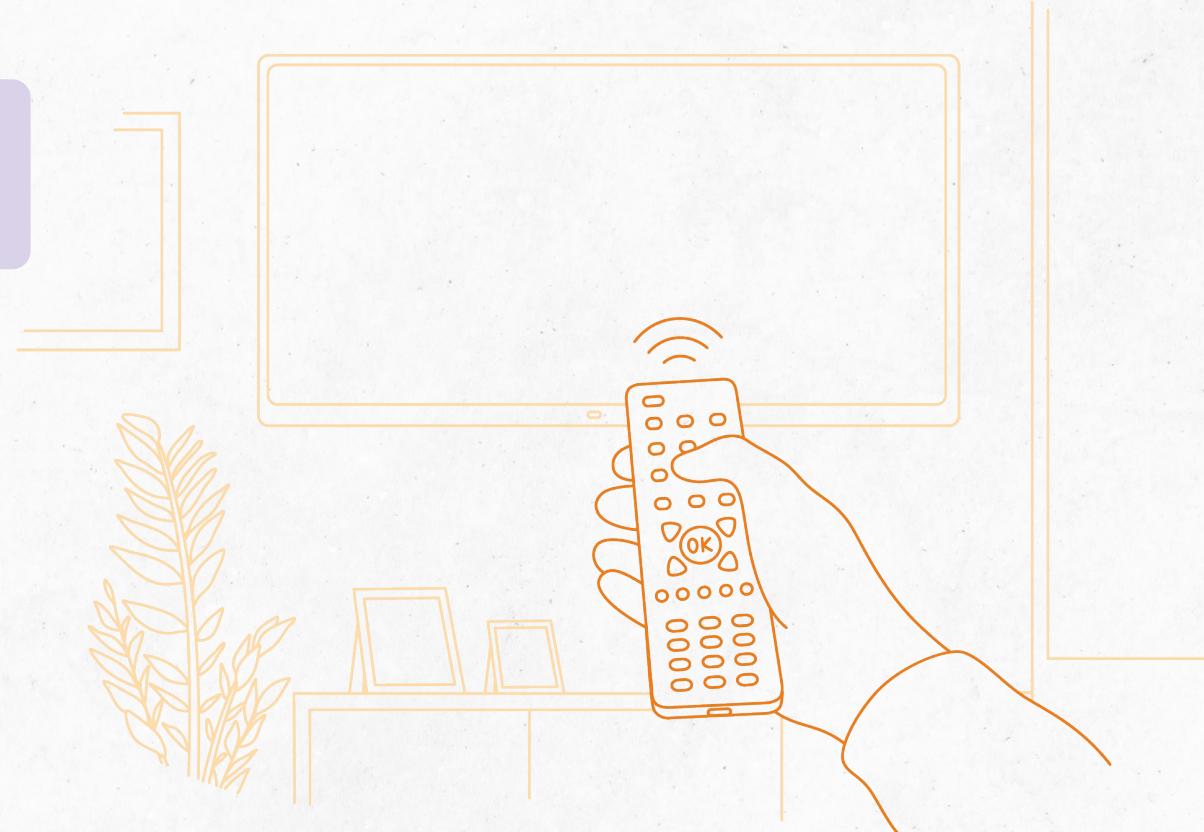
Validação: como ele foi testado?

Evolução: como ele evoluiu ao longo do tempo?



Fechamento

Se a aula de hoje fosse um filme,
qual seria e por quê?



Referências Bibliográficas

PROZ EDUCAÇÃO. *Qualidade de software*. 2023.