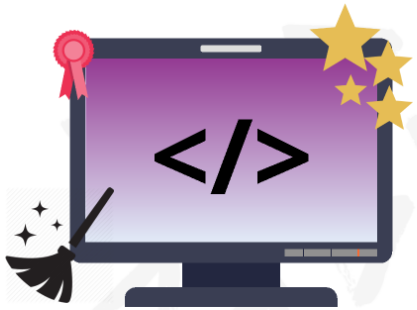
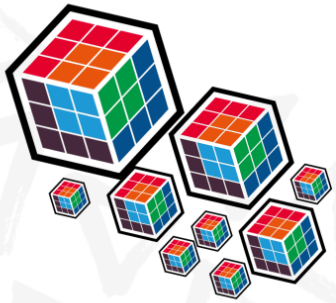
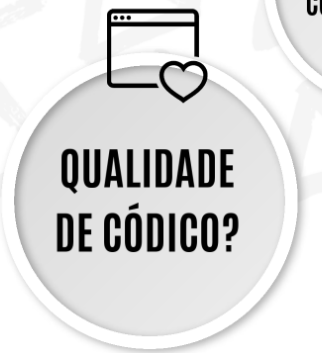
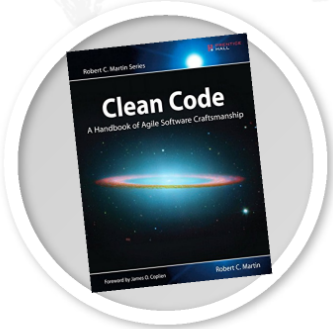









# Codificando funcionalidade e produtividade sem medo





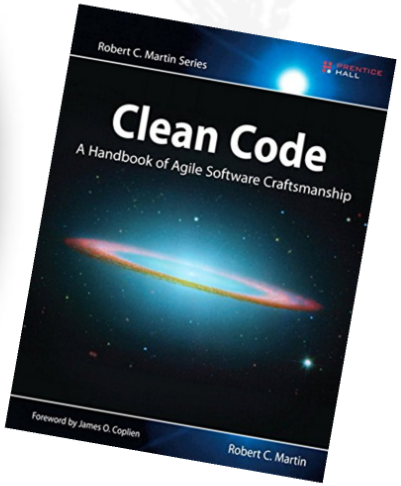
Lorena Dutra
Bacharelado em Sistemas de Informação - UFSM
Mestrado em ciência da Computação - UFSM
Desenvolvedora - Softplan

THE DEVELOPER'S CONFERENCE

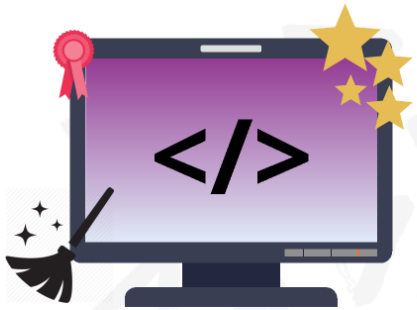
“ Aprender a criar códigos limpos é uma tarefa árdua e requer mais do que o simples conhecimento dos princípios e padrões.

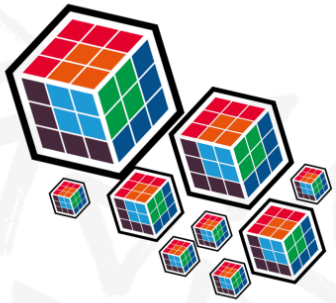
Você deve suar a camisa;
Praticar sozinho e ver que cometeu erros;
Assistir os outros praticarem e errarem;
Vê-los tropeçar e refazer seus passos;
Entender o preço a ser pago por cada decisão tomada de maneira errada. ”

*Robert C. Martin*



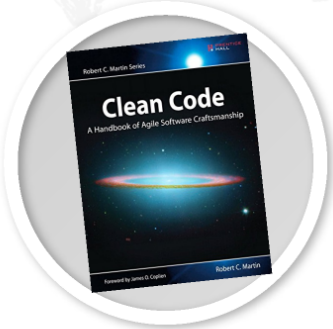

THE DEVELOPER'S CONFERENCE



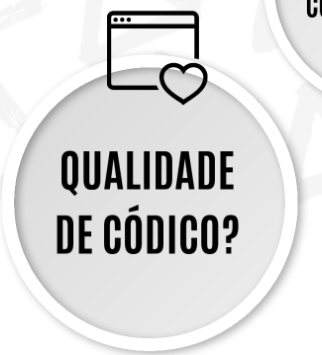




# Codificando funcionalidade e produtividade sem medo

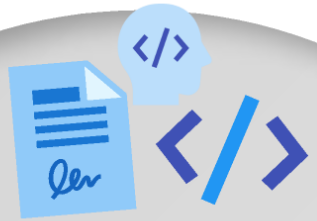







Lorena Dutra
Bacharelado em Sistemas de Informação - UFSM
Mestrado em ciência da Computação - UFSM
Desenvolvedora - Softplan

THE DEVELOPER'S CONFERENCE



A produtividade de um programador está diretamente ligada a qualidade do código em que ele trabalha.

Realizar a manutenção de um software é bem mais complicado do que desenvolver um do início.

THE DEVELOPER'S CONFERENCE

Um código bem escrito é um diferencial em qualquer projeto, nesse sentido, um bom desenvolvedor deve se preocupar em **entregar valor e qualidade** aos seus projetos, independente da linguagem, padrão ou frameworks utilizados.

Saber seguir o caminho certo para a resolução de diferentes problemas é fundamental para o desenvolvimento e aprimoramento de um projeto.

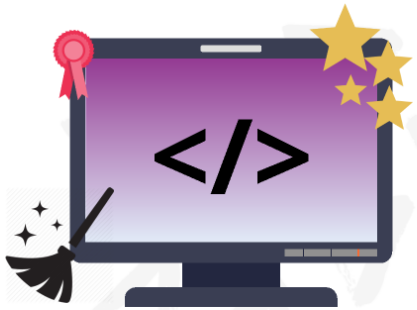
# DESIGN DE CÓDIGO

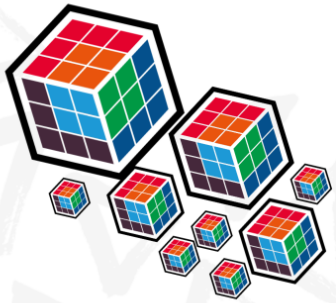
- Tornar a vida do usuário mais fácil.
  - Pensar no conforto que alguém virá a ter ao ler determinado código.
    - Código fonte limpo.
      Fácil de entender.
    - Elevando a compreensão e melhorando a funcionalidade.

# DESIGN DE CÓDIGO

Revelando a sua real intenção.

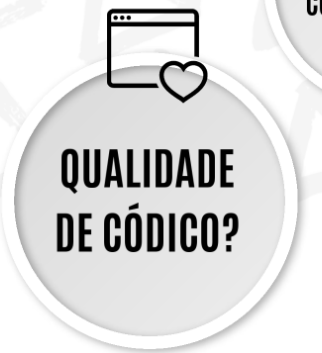
- Tornar a vida do usuário mais fácil.
  - Pensar no conforto que alguém virá a ter ao ler determinado código.
    - Código fonte limpo.
      Fácil de entender.
    - Elevando a compreensão e melhorando a funcionalidade.



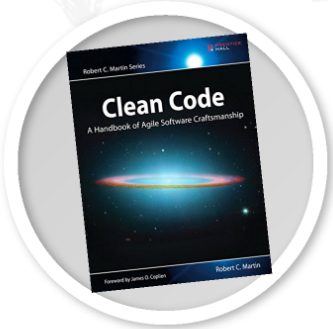









# Codificando funcionalidade e produtividade sem medo





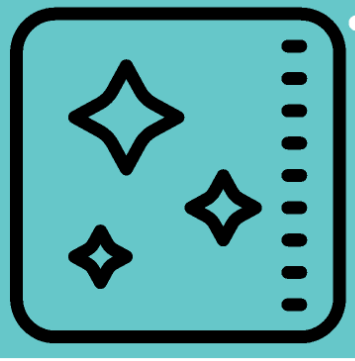
Lorena Dutra
Bacharelado em Sistemas de Informação - UFSM
Mestrado em ciência da Computação - UFSM
Desenvolvedora - Softplan


THE DEVELOPER'S CONFERENCE

# Motivação

Se o código não estiver limpo, o desenvolvimento decai com o tempo.

- Incerteza
- Tempo gasto
- Frustração
- Decepção
- Reescrita
- Dor



Simplicidade

Ausência de duplicidade

Facilidade de leitura

Elegância

# Nomes

Devem descrever objetivos

1

2

3

4

# Nomes

Devem descrever objetivos

Por que ele existe?

1

2

3

4

# Nomes

Devem descrever objetivos

- Por que ele existe?
- O que ele faz?

1

2

3

4

# Nomes

Devem descrever objetivos

- Por que ele existe?
- O que ele faz?
- Como ele é usado?

1

2

3

4

Deve eliminar a necessidade de um comentário de explicação.

# Deve eliminar a necessidade de um comentário de explicação.

```
private DateTime dnDt; // Data de Nascimento
private int buscarIdade() {
    return (int)(DateTime.Now().Subtract(dnDt).TotalDays / 365);
}
```

# Deve eliminar a necessidade de um comentário de explicação.

```
private DateTime dnDt; // Data de Nascimento
private int buscarIdade() {
    return (int)(DateTime.Now().Subtract(dnDt).TotalDays / 365);
}
```

# Deve eliminar a necessidade de um comentário de explicação.

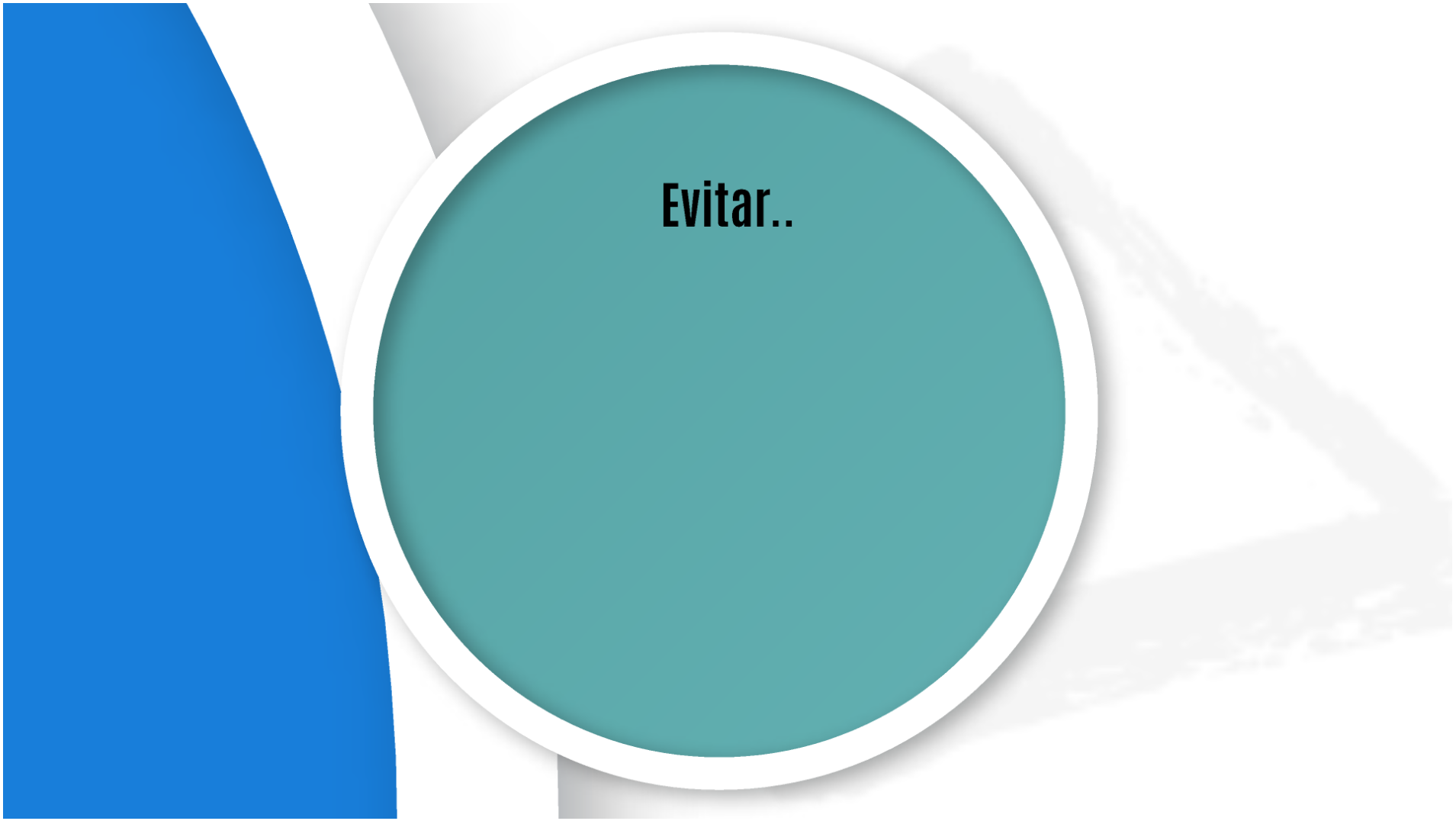
```
private DateTime dnDt; // Data de Nascimento
private int buscarIdade() {
    return (int)(DateTime.Now().Subtract(dnDt).TotalDays / 365);
}
```

```
private DateTime dataNascimento;
private const DIAS_POR_ANO = 365;
private int buscarIdade() {
    int diasDiferenca = DateTime.Now().Subtract(dataNascimento).TotalDays;
    return (int) (diasDiferenca / DIAS_POR_ANO);
}
```

# Deve eliminar a necessidade de um comentário de explicação.

```
private DateTime dnDt; // Data de Nascimento
private int buscarIdade() {
    return (int)(DateTime.Now().Subtract(dnDt).TotalDays / 365);
}
```

```
private DateTime dataNascimento;
private const DIAS_POR_ANO = 365;
private int buscarIdade() {
    int diasDiferenca = DateTime.Now().Subtract(dataNascimento).TotalDays;
    return (int) (diasDiferenca / DIAS_POR_ANO);
}
```


Evitar..

# Evitar..

Nomes com o tipo na descrição - *ArrayContatos;*

L maiúsculo ou minusculos - podem ser confundidos com o número 1;

Nomes não pronunciáveis;

# Evitar..

Nomes com o tipo na descrição - *ArrayContatos;*

L maiúsculo ou minusculos - podem ser confundidos com o número 1;

Nomes não pronunciáveis;

Nomes com apenas uma letra ou numéricos

# Nomes com apenas uma letra ou numéricos

```
for (int j = 0; j < 34; j++) {
    a += (t[j]*4)/5;
}
```

# Nomes com apenas uma letra ou numéricos

```
for (int j = 0; j < 34; j++) {
    a += (t[j]*4)/5;
}
```

# Nomes de métodos

Boas Práticas!

# Nomes de métodos

Boas Práticas!

```
UsuarioService.buscar(18);
```

# Nomes de métodos

Boas Práticas!

```
UsuarioService.buscar(18);
```

```
// Retorna uma lista de usuários pela idade
public List<Usuario> buscar(int idade)
{
    return buscarTodosUsuarios().Where(q => q.Idade == idade);
}
```

# Nomes de métodos

Boas Práticas!

```
UsuarioService.buscar(18);
```

```
// Retorna uma lista de usuários pela idade
public List<Usuario> buscar(int idade)
{
    return buscarTodosUsuarios().Where(q => q.Idade == idade);
}
```

# Nomes de métodos

Boas Práticas!

```
UsuarioService.buscar(18);
```

```
// Retorna uma lista de usuários pela idade
public List<Usuario> buscar(int idade)
{
    return buscarTodosUsuarios().Where(q => q.Idade == idade);
}
```

```
UsuarioService.buscarPorIdade(18);
```

# Nomes de métodos

Boas Práticas!

```
UsuarioService.buscar(18);
```

```
// Retorna uma lista de usuários pela idade
public List<Usuario> buscar(int idade)
{
    return buscarTodosUsuarios().Where(q => q.Idade == idade);
}
```

```
UsuarioService.buscarPorIdade(18);
```

```
public List<Usuario> buscarPorIdade(int idade)
{
    return buscarTodosUsuarios().Where(q => q.Idade == idade);
}
```

# Nomes de métodos

Boas Práticas!

```
UsuarioService.buscar(18);
```

```
// Retorna uma lista de usuários pela idade
public List<Usuario> buscar(int idade)
{
    return buscarTodosUsuarios().Where(q => q.Idade == idade);
}
```

```
UsuarioService.buscarPorIdade(18);
```

```
public List<Usuario> buscarPorIdade(int idade)
{
    return buscarTodosUsuarios().Where(q => q.Idade == idade);
}
```

# Boas Práticas!

# Boas Práticas!

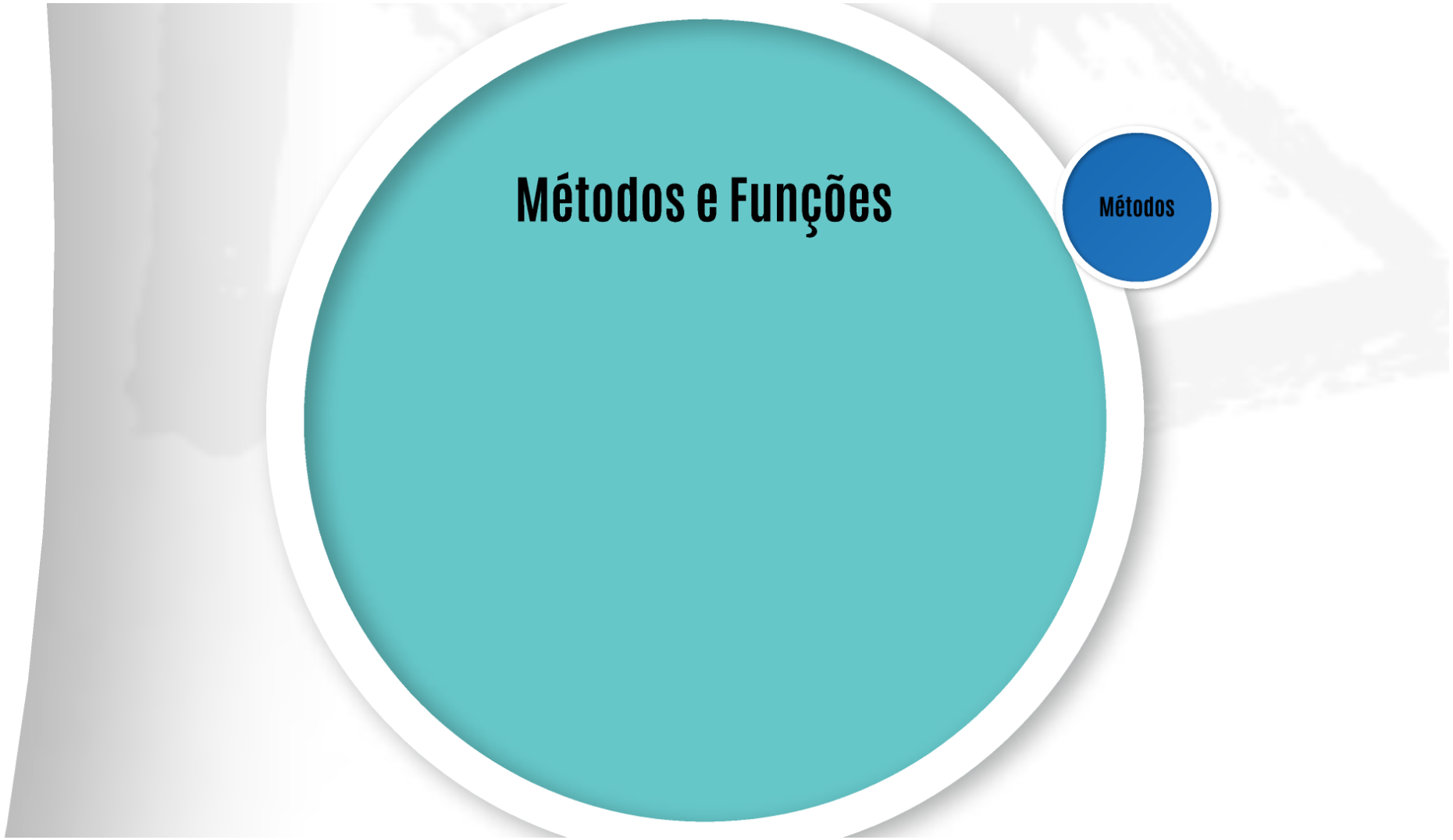
# Boas Práticas!

Nomes de classes devem ser substantivos e não conter verbos;

# Boas Práticas!

Nomes de classes devem ser substantivos e não conter verbos;

Nomes de métodos devem conter verbos.


Métodos e Funções
Métodos

# Métodos e Funções

Métodos

# Métodos e Funções

Métodos

Linha de organização de qualquer software

# Métodos e Funções

Métodos

1. Linha de organização de qualquer software

- O que faz um método fácil de ler?

# Métodos e Funções

Métodos

1. Linha de organização de qualquer software

- O que faz um método fácil de ler?
- Como fazer que um método transmita objetividade?

# Métodos e Funções

Métodos

1. Linha de organização de qualquer software

- O que faz um método fácil de ler?
- Como fazer que um método transmita objetividade?
- Quais atributos passar como parâmetro?

# Boas Práticas

Tamanho

- Nomes de classes = Substantivos, não verbos.
- Nomes de métodos = Verbos.
- Métodos devem ser pequenos.

# Boas Práticas

Tamanho

- Nomes de classes = Substantivos, não verbos.
- Nomes de métodos = Verbos.
- Métodos devem ser pequenos.

**Como Mensurar?**

# Boas Práticas

Tamanho

- Nomes de classes = Substantivos, não verbos.
- Nomes de métodos = Verbos.
- Métodos devem ser pequenos.

**Como Mensurar?**

# Tamanho dos métodos

# Tamanho dos métodos

Até 20 linhas

# Tamanho dos métodos

Até 20 linhas

Linhas com menos de 100 caracteres

# Tamanho dos métodos

Até 20 linhas

Linhas com menos de 100 caracteres

Identação < 2

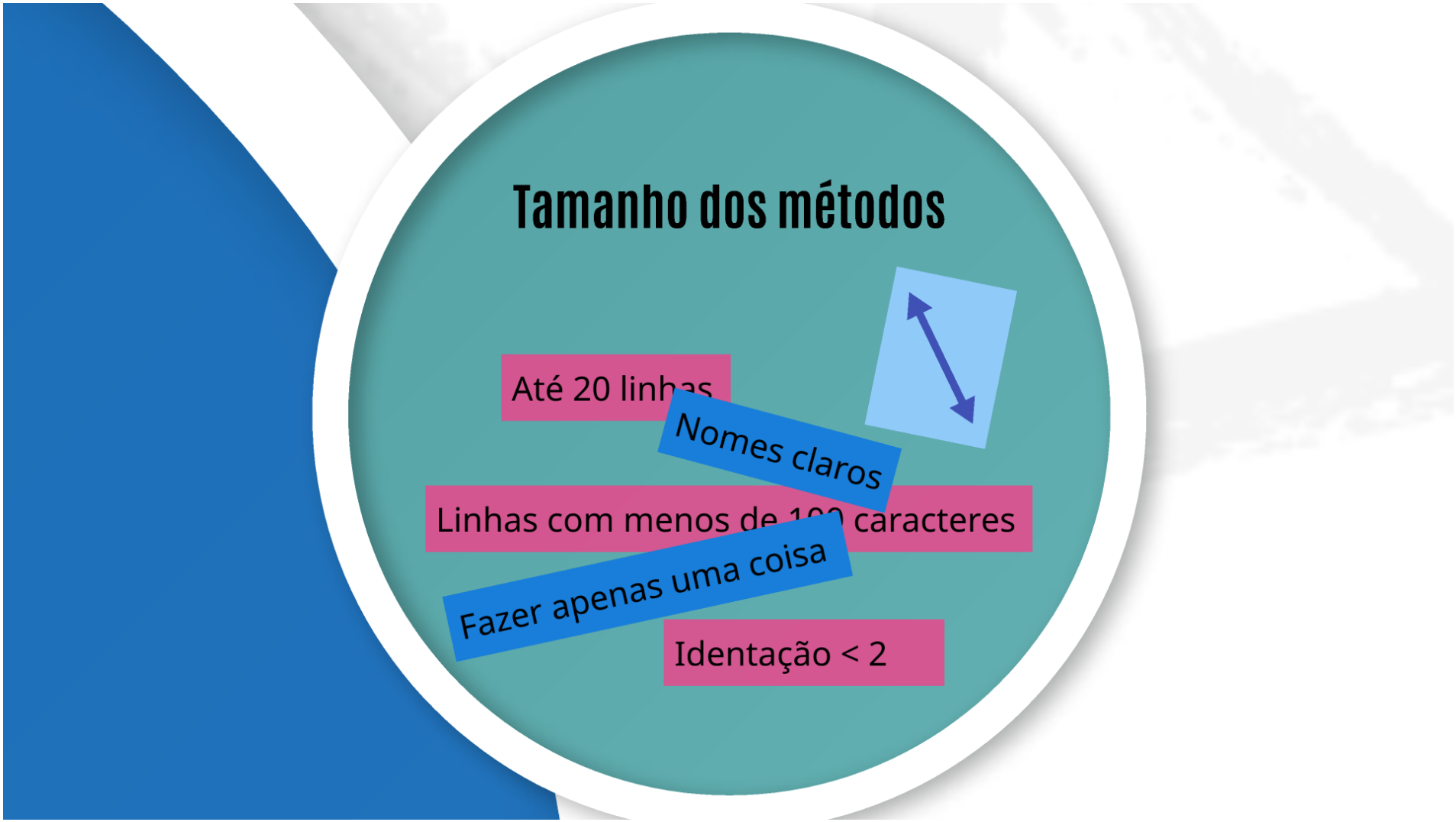
# Tamanho dos métodos

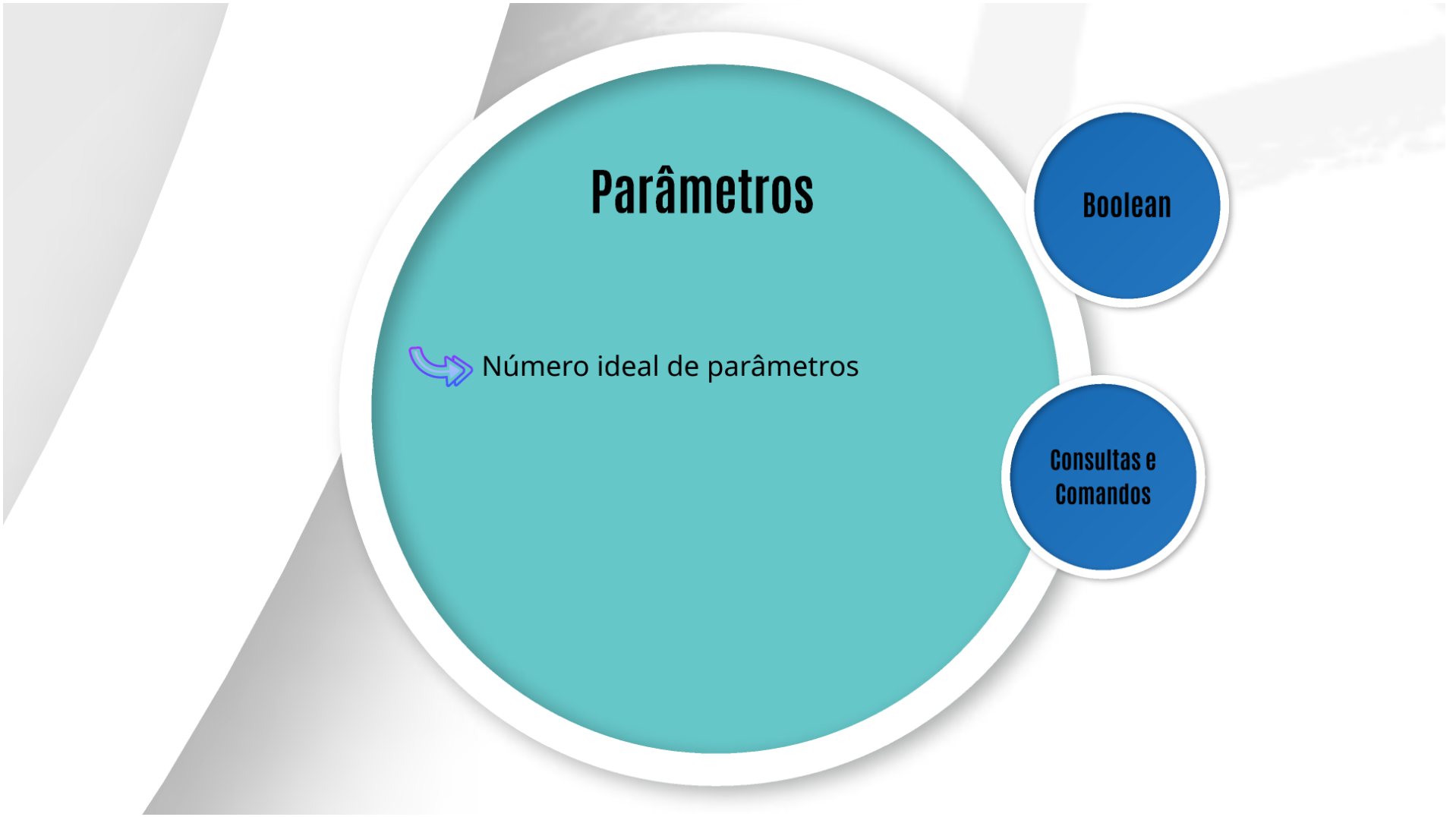
Até 20 linhas

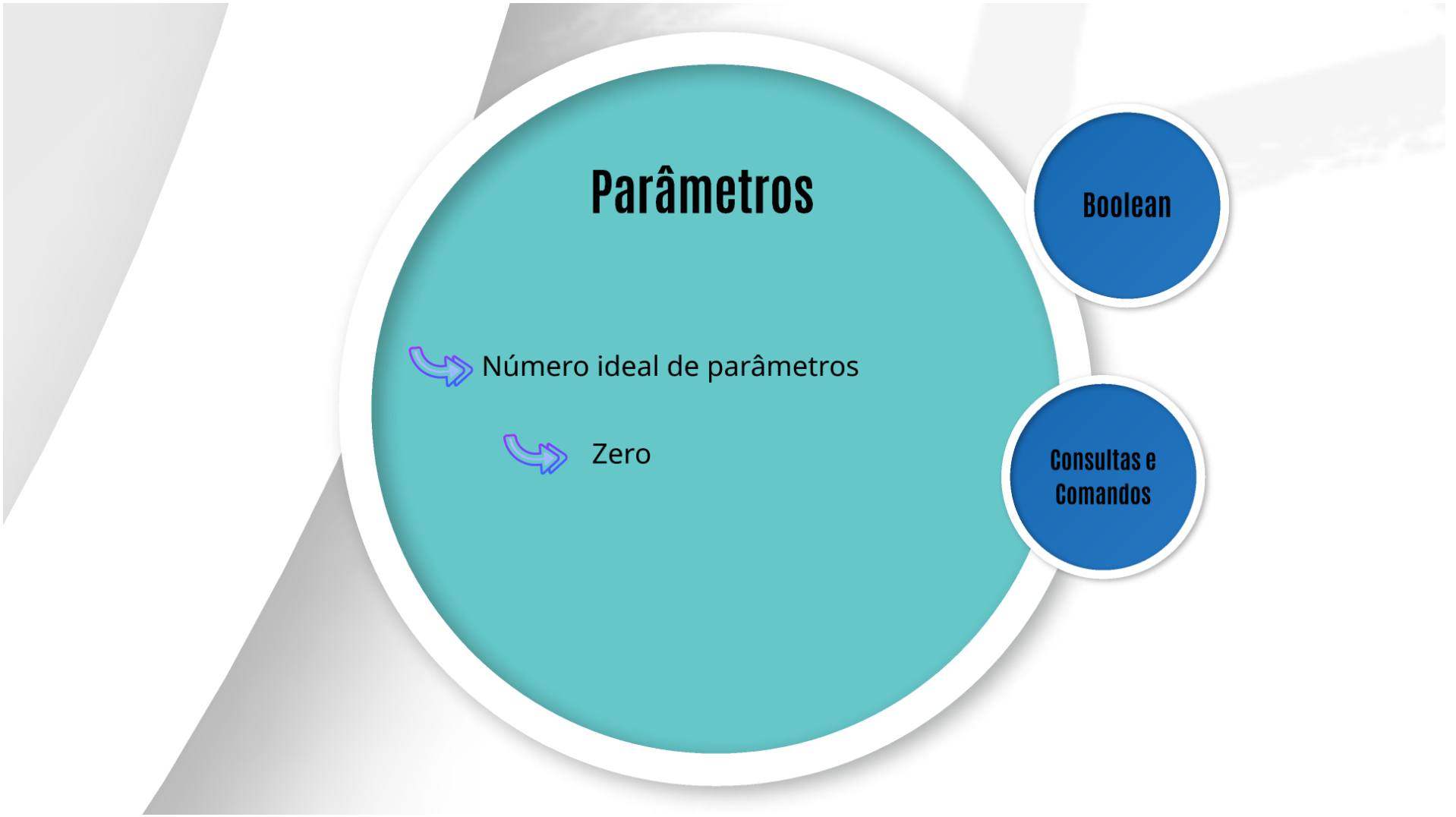
Linhas com menos de 100 caracteres

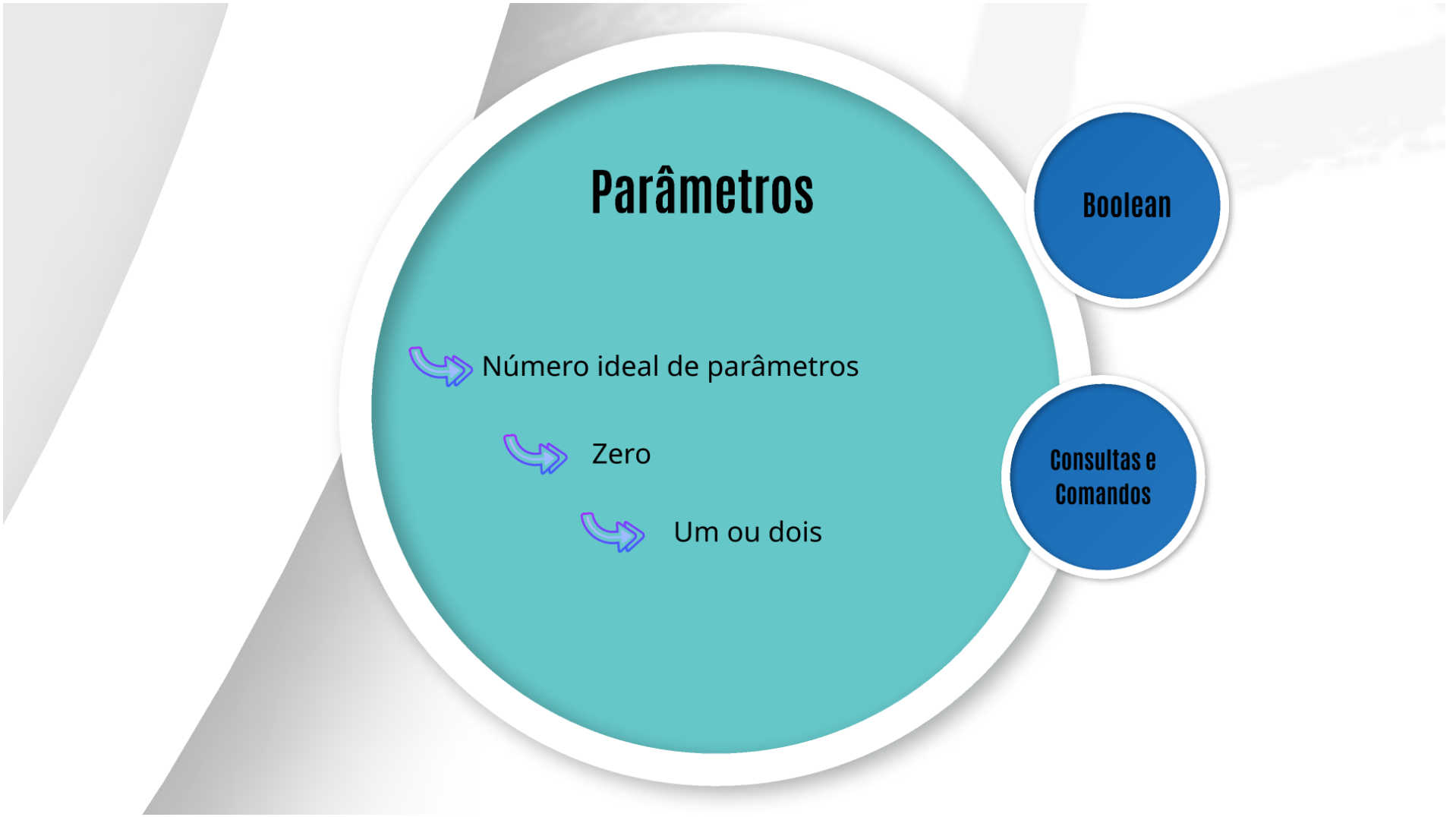
Fazer apenas uma coisa

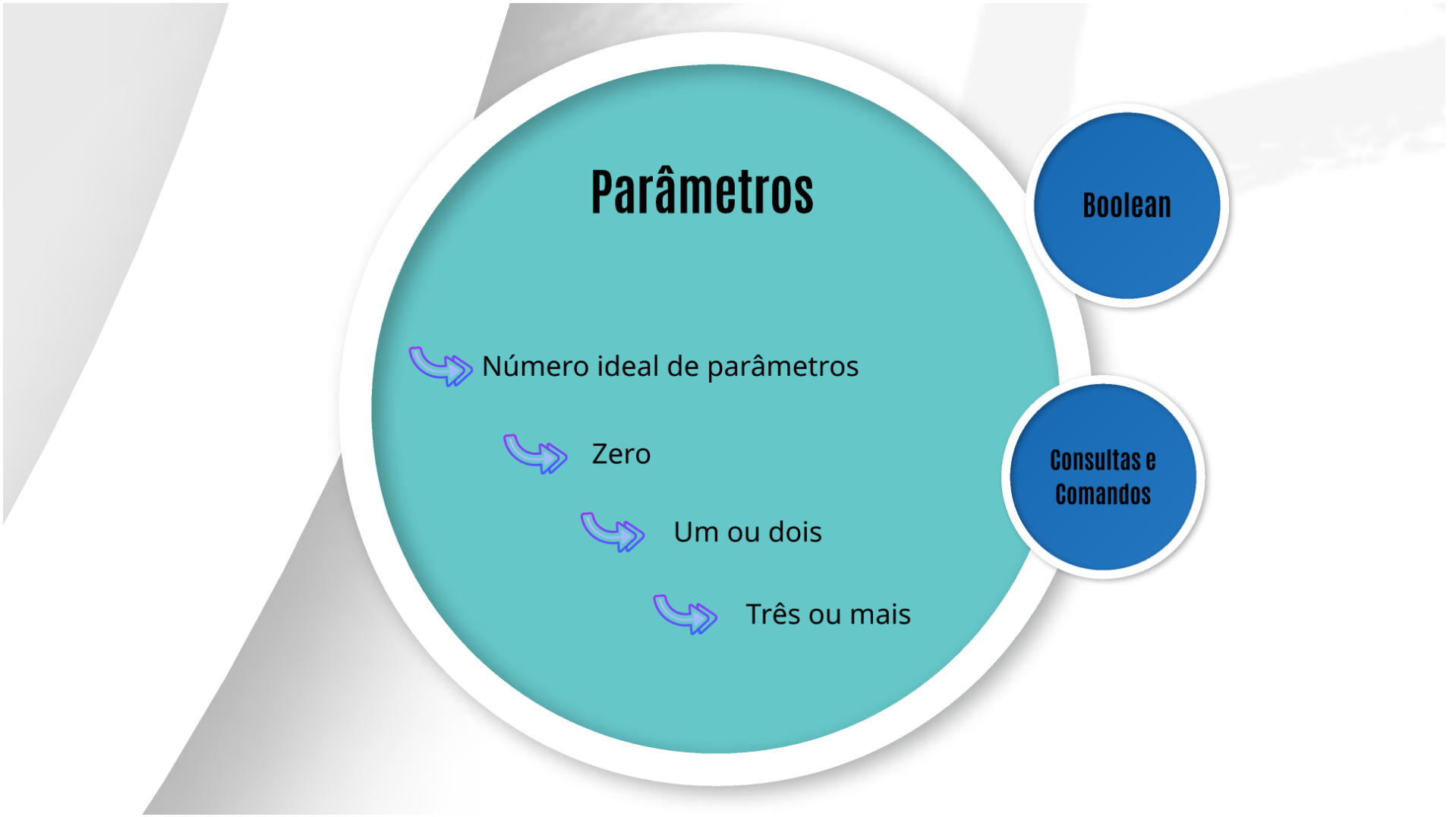
Identação < 2

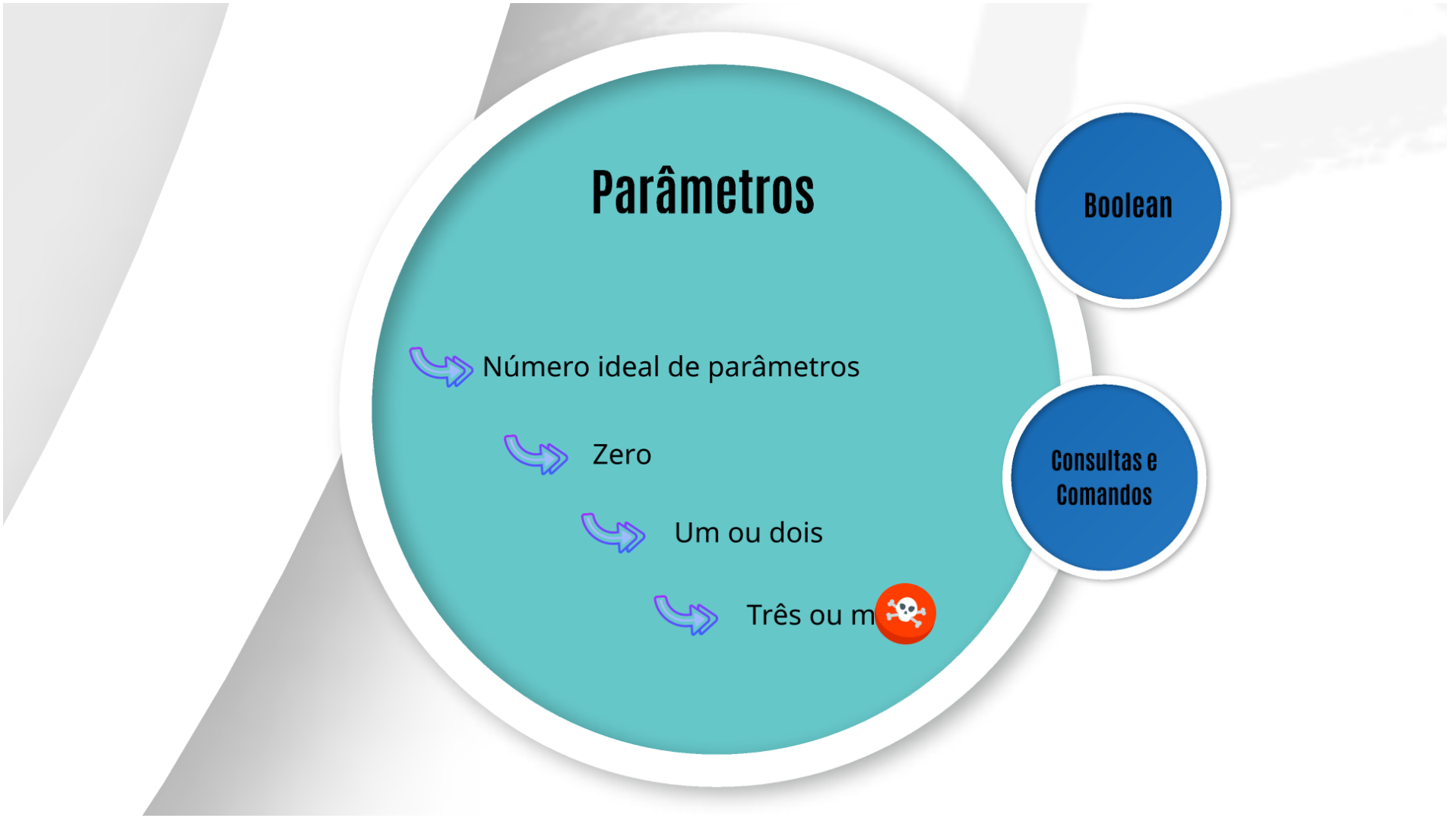

Tamanho dos métodos
Até 20 linhas
Nomes claros
Linhas com menos de 100 caracteres
Fazer apenas uma coisa
Identação < 2


Parâmetros
Número ideal de parâmetros
Boolean
Consultas e Comandos


Parâmetros
Número ideal de parâmetros
Zero
Boolean
Consultas e Comandos


Parâmetros
Número ideal de parâmetros
Zero
Um ou dois
Boolean
Consultas e Comandos


Parâmetros
Número ideal de parâmetros
Zero
Um ou dois
Três ou mais
Boolean
Consultas e Comandos


Parâmetros
Número ideal de parâmetros
Zero
Um ou dois
Três ou m
Boolean
Consultas e Comandos

# Parâmetros do tipo Boolean

# Parâmetros do tipo Boolean

Deve ser evitada!

# Parâmetros do tipo Boolean

Deve ser evitada!

Está dizendo que a função faz mais de uma coisa.

# Consultas e Comandos

Métodos fazem alguma coisa ou retornam alguma coisa

# Consultas e Comandos

Métodos fazem alguma coisa ou retornam alguma coisa

# Comentários

# Comentários

# Comentários

- Úteis, se colocados em lugares corretos

# Comentários

- Úteis, se colocados em lugares corretos
- Podem ser mentirosos

# Comentários

- Úteis, se colocados em lugares corretos
- Podem ser mentirosos

Só o código pode dizer o que ele realmente faz

# Comentários

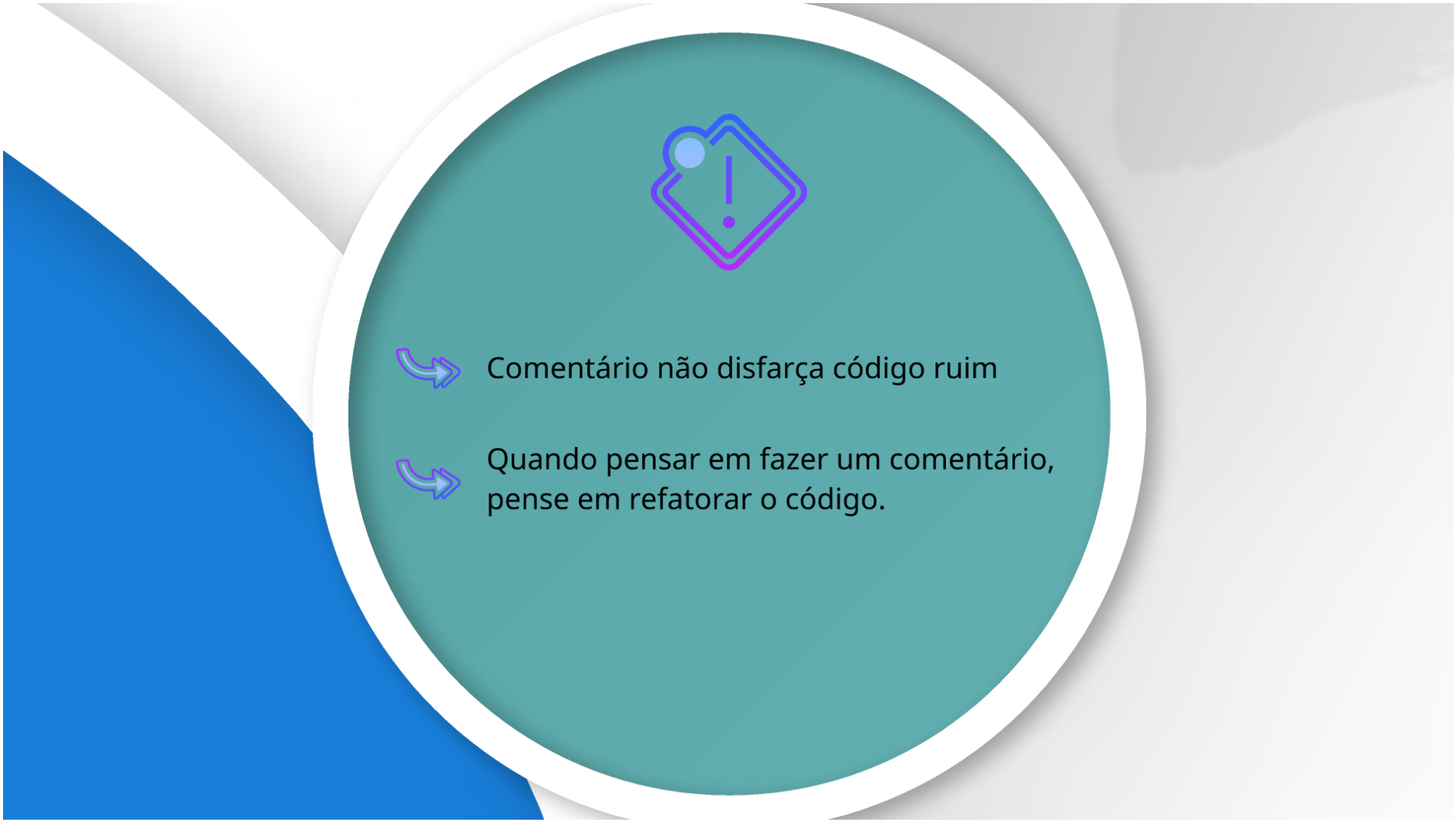
Úteis, se colocados em lugares adequados

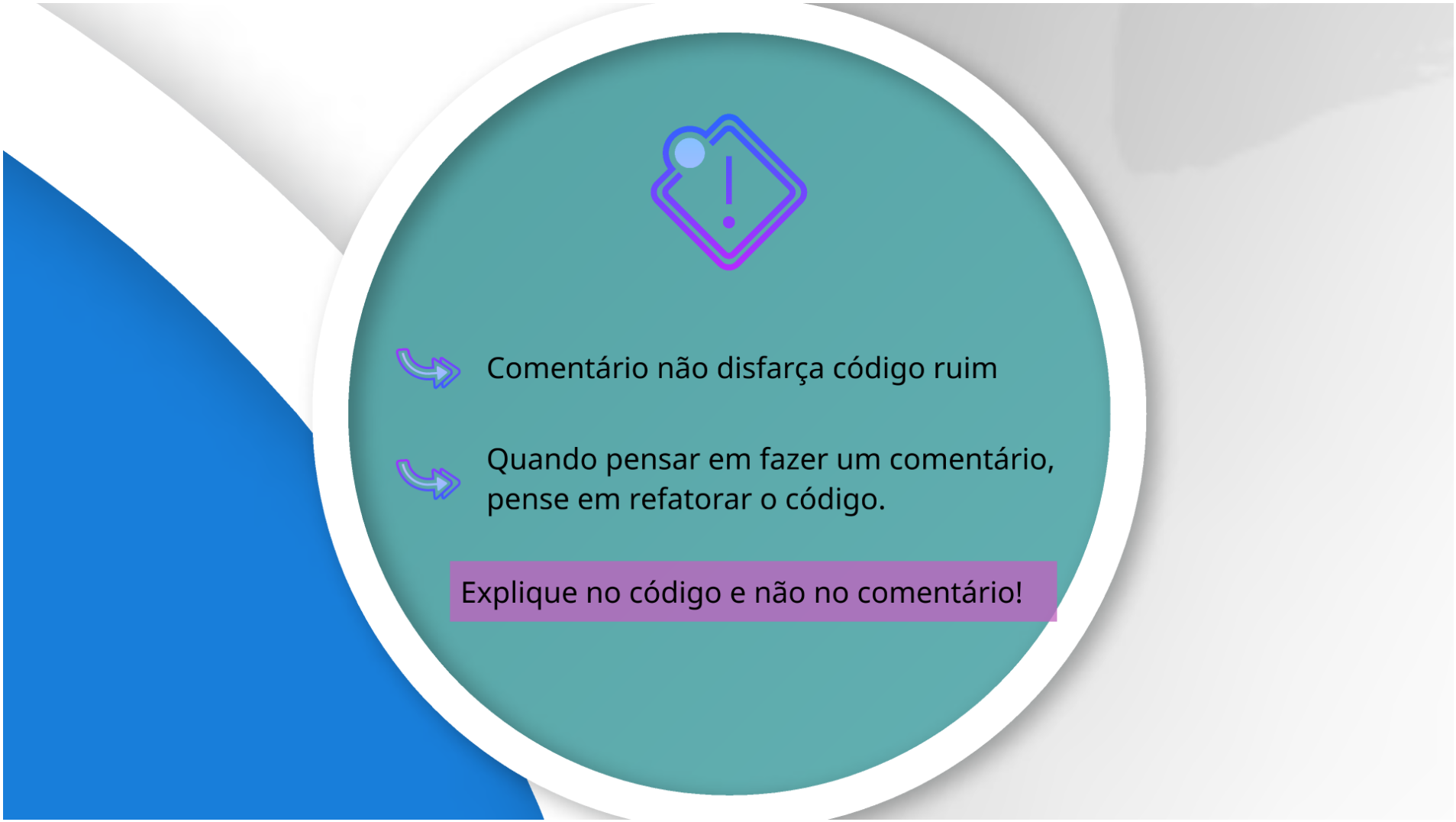
Comentários não recebem manutenção!

Podem ser mentirosos

Só o código pode dizer o que ele realmente faz


Comentário não disfarça código ruim

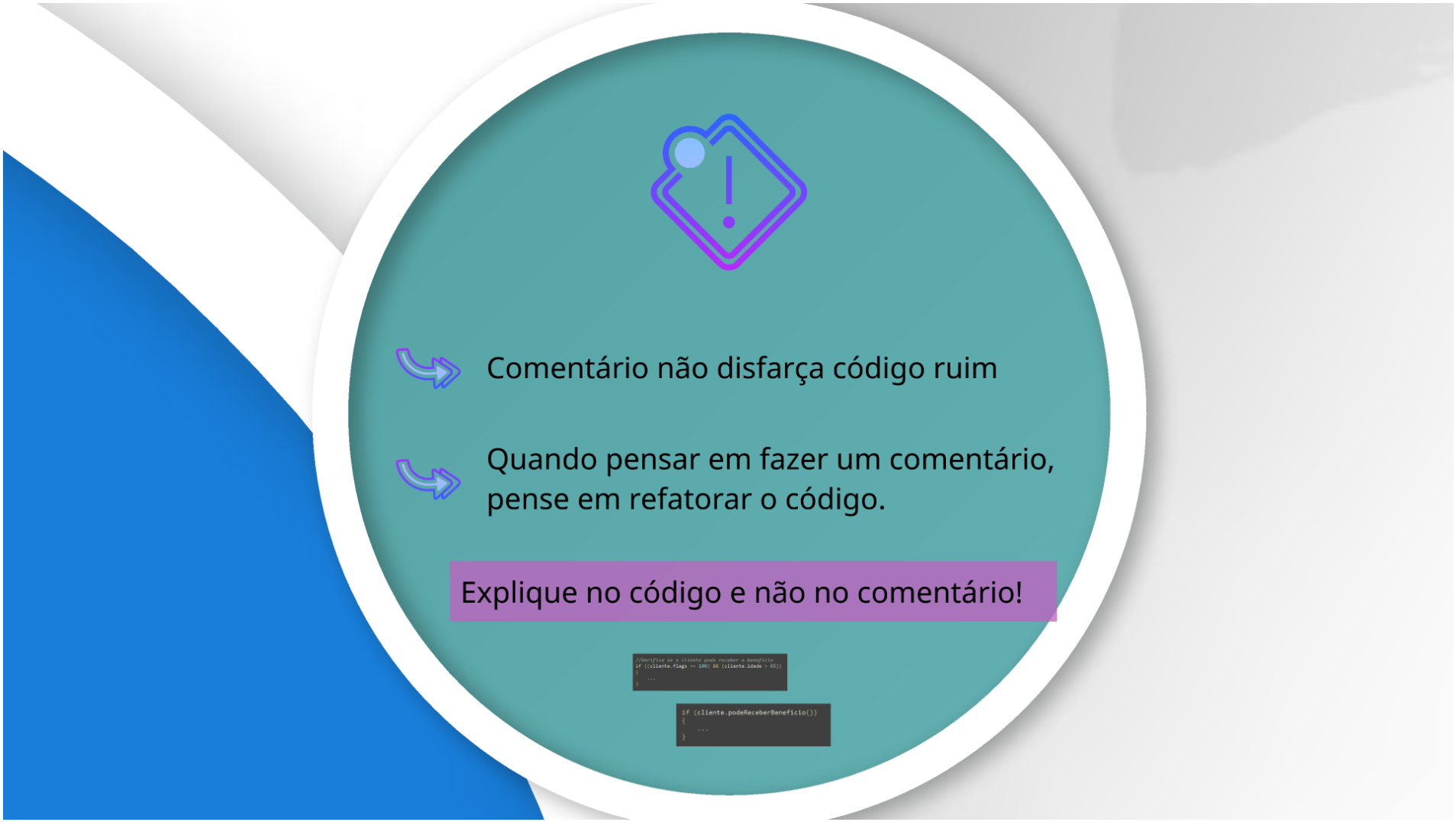

Comentário não disfarça código ruim
Quando pensar em fazer um comentário, pense em refatorar o código.


Comentário não disfarça código ruim
Quando pensar em fazer um comentário, pense em refatorar o código.
Explique no código e não no comentário!

```
//Verifica se o cliente pode receber o benefício
if ((cliente.flags == 100) && (cliente.idade > 65))
{
    ...
}
```

```
//Verifica se o cliente pode receber o benefício
if ((cliente.flags == 100) && (cliente.idade > 65))
{
    ...
}
```

```
if (cliente.podeReceberBeneficio())
{
    ...
}
```



- Comentário não disfarça código ruim
- Quando pensar em fazer um comentário, pense em refatorar o código.

Explique no código e não no comentário!

```
//Verifica se o cliente pode receber o benefício
if ((cliente.flags == 100) && (cliente.idade > 65))
{
    ...
}
```

```
if (cliente.podeReceberBeneficio())
{
    ...
}
```

# Tipos de comentários

**Bons:** Necessário ou benéficos.

**Explicativo:** Fornece a intenção por trás de uma decisão tomada.

**Alerta:** Avisar sobre algumas consequências.

**Ruins:** Forçam você a olhar em outra parte do código para entendê-lo.

**Redundantes:** Não diz mais que o próprio código, declaram o óbvio.

**Enganoso:** Comentário que não é exato.

**Confuso:** Quando o comentário precisa ser explicado.

Fechamento

# Comentários de fechamentos


Se eles existem, quer dizer que o código está muito grande.

```
try {
    string nada;
    while ( i + 3) {
        i++;
        ...
        ...
    } // end while
} // end try
```


Código comentado
Exemplos

# Código comentado

Exemplos

```
// NUNCA FAÇA ISSO
```

# Código comentado

Exemplos

```
// NUNCA FAÇA ISSO
```

Quando alguém vê um código comentado não tem coragem de apagá-lo.

Incerteza

Existe algum motivo para estar ali ?

# Exemplos

Tomara que eu nunca encontre estes tais conjuntos:

```
//tem esse "for" para o tratamento do malditos conjuntos
```

Trocando 6 por meia dúzia:

```
/isso ta horrivel, temos que pensar como colocar isso dentro da lib de um modo genérico
```

Consciência pesada?

```
//melhorar isso
```

Conhecimento de causa:

```
//Evitando o php MALUCO
```

Identação

# Formatação

Importante porque é o meio de comunicação.

Primeira ordem para os desenvolvedores.

A legibilidade do seu código terá profundo efeito todas as mudanças que serão feitas.

# Identação

Uma boa identação do código ajuda a visualizar todo o escopo.

Identificar as situações e regras relevantes mais rápido.

```
public boolean checkPassword(String username, String password) {
String passwordStatus = cryptographer.descrypt(password);
if(passwordStatus.equals("Ok")) {
return true;
}
return false;
}
```

```
public boolean checkPassword(String username, String password)
{
    String passwordStatus = cryptographer.descrypt(password);
    if(passwordStatus.equals("Ok"))
        return true;
    else
        return false;
}
```

# relevantes mais rapido.

```
public boolean checkPassword(String username, String password) {
String passwordStatus = cryptographer.descrypt(password);
if(passwordStatus.equals("Ok")) {
return true;
}
return false;
}
```

```
public boolean checkPassword(String username, String password)
{
    String passwordStatus = cryptographer.descrypt(password);
    if(passwordStatus.equals("Ok"))
        return true;
    else
        return false;
}
```

# Identação

Uma boa identação do código ajuda a visualizar todo o escopo.

Identificar as situações e regras relevantes mais rápido.

```
public boolean checkPassword(String username, String password) {
String passwordStatus = cryptographer.descrypt(password);
if(passwordStatus.equals("Ok")) {
return true;
}
return false;
}
```

```
public boolean checkPassword(String username, String password)
{
    String passwordStatus = cryptographer.descrypt(password);
    if(passwordStatus.equals("Ok"))
        return true;
    else
        return false;
}
```

# Erros

Tratar erros é uma das coisas que todos nós temos que fazer quando estamos programando.

As coisas podem dar errado e nós temos que estar certos que nosso código fará o que deve fazer.

Exceptions

Contexto

# Exceptions

Utilize exceptions ao invés de retornar código de erro.

O problema desses retornos é que eles desorganizam a chamada.

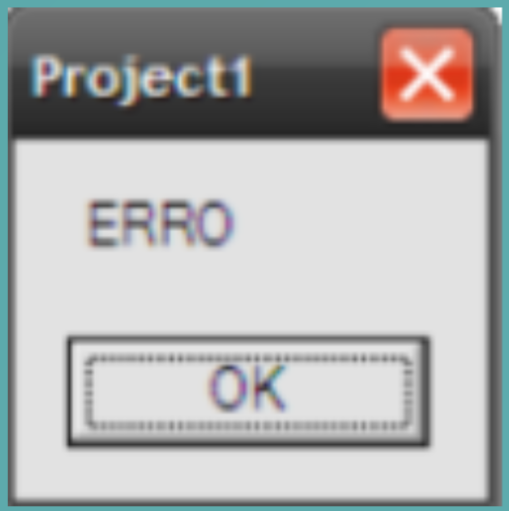
Quem fez a chamada deve verificar se há erros no retorno e isso pode ser fácil de esquecer.

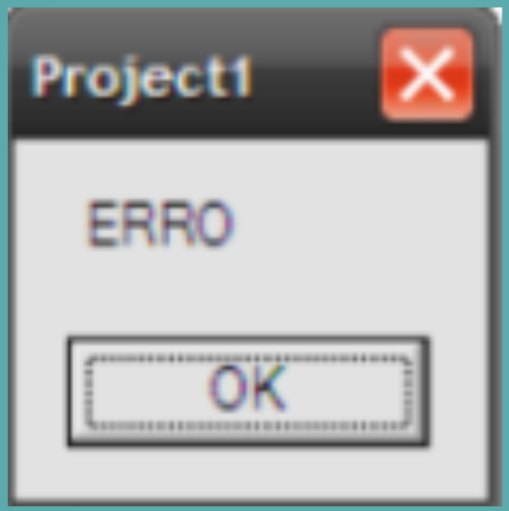
Por isso que é melhor lançar uma exception

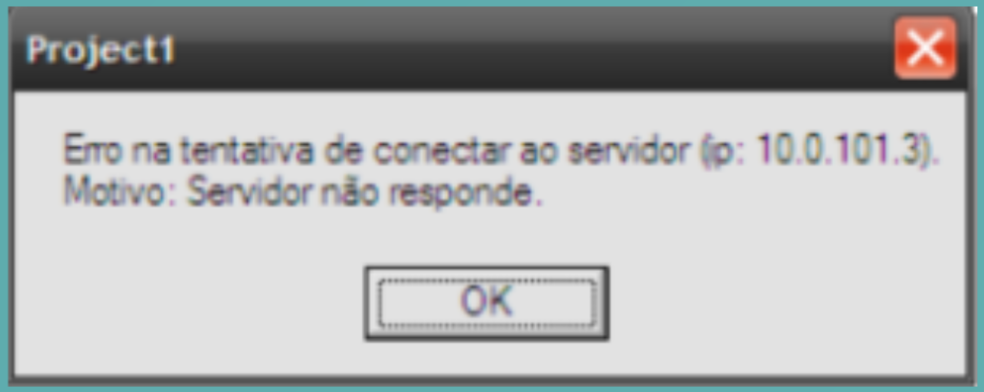
# Forneca o contexto

Crie mensagens informativas para os erros.

Mencione a operação que falhou e o tipo de falha.


Project1
ERRO
OK

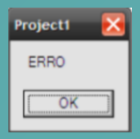

Project1
ERRO
OK

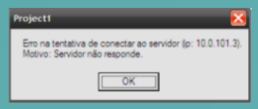

Project1
Erro na tentativa de conectar ao servidor (ip: 10.0.101.3).
Motivo: Servidor não responde.
OK

# Forneca o contexto

Crie mensagens informativas para os erros.

Mencione a operação que falhou e o tipo de falha.



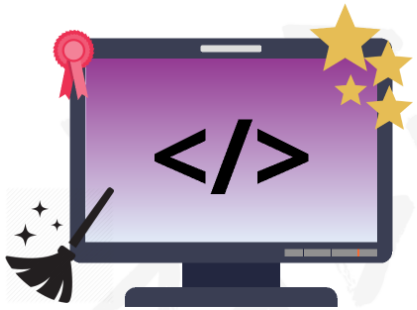



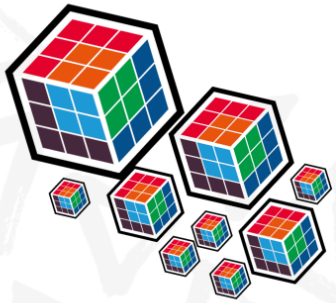
Torne escrever código limpo parte do seu processo pessoal.

Aprenda como escrever código limpo.

Escrever código limpo é como realizar qualquer outro tipo de trabalho.

Requer prática, criticismo e mais prática.





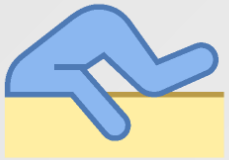


# Codificando funcionalidade e produtividade sem medo



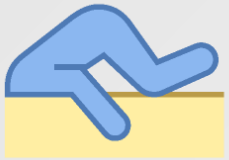



Lorena Dutra
Bacharelado em Sistemas de Informação - UFSM
Mestrado em ciência da Computação - UFSM
Desenvolvedora - Softplan



- Para entregar resultados muito rápidos.
- "Saco cheio" de alguma tarefa e concluir de qualquer forma somente para pegar uma próxima tarefa mais interessante.
- Os requisitos mudaram ao longo do caminho.

THE DEVELOPER'S CONFERENCE

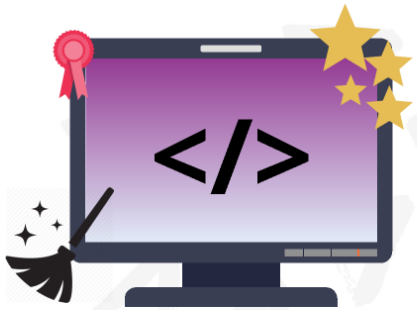


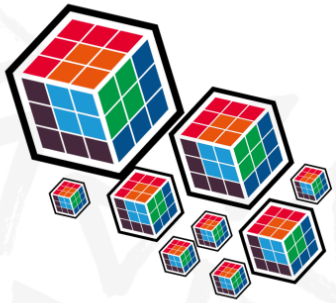
- Para entregar resultados muito rápidos

Caso você fosse um médico e seu paciente (apressado) pedisse para que você não lavasse as mãos para que a cirurgia fosse mais rápida, você iria obedecer?

- Os requisitos mudaram ao longo do caminho.

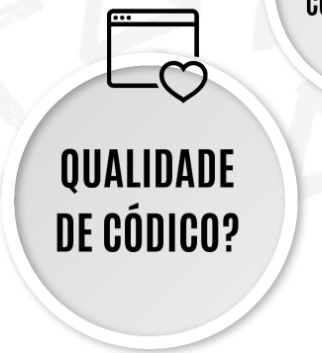
THE DEVELOPER'S CONFERENCE



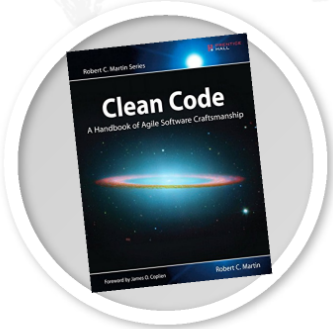






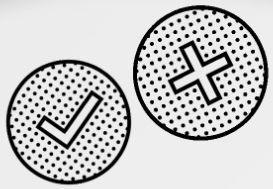



# Codificando funcionalidade e produtividade sem medo



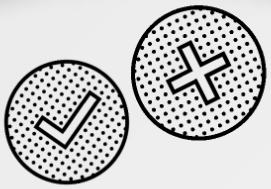


Lorena Dutra
Bacharelado em Sistemas de Informação - UFSM
Mestrado em ciência da Computação - UFSM
Desenvolvedora - Softplan



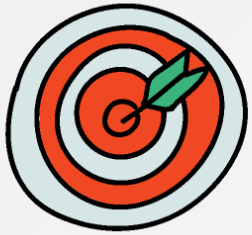
Aumenta o tempo de desenvolvimento de um projeto de software ou acelera ?

THE DEVELOPER'S CONFERENCE

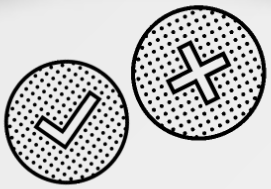


Aumenta o tempo de desenvolvimento de um projeto de software ou acelera ?

Produtividade

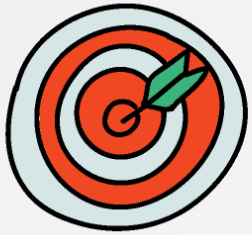


THE DEVELOPER'S CONFERENCE



Aumenta o tempo de desenvolvimento de um projeto de software ou acelera ?

Produtividade



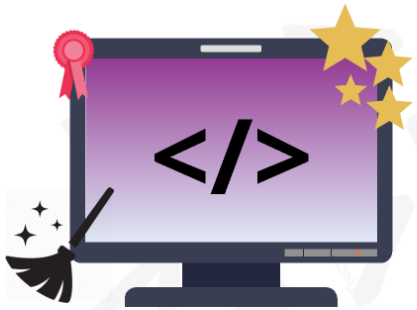
"Se eles estão juntos e concentrados trabalhando, deve ser algo importante, melhor deixar a conversa para outra hora!".

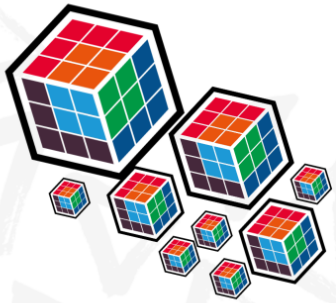
THE DEVELOPER'S CONFERENCE

Programação em pares apresenta resultados positivos quando pensamos em qualidade.

“Duas cabeças pensam melhor que uma”,

Evolução no nível técnico e motivacional das equipes .







# Codificando funcionalidade e produtividade sem medo

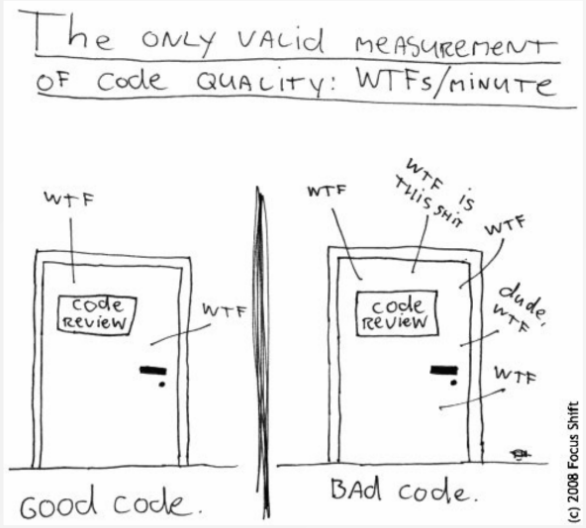





Lorena Dutra
Bacharelado em Sistemas de Informação - UFSM
Mestrado em ciência da Computação - UFSM
Desenvolvedora - Softplan

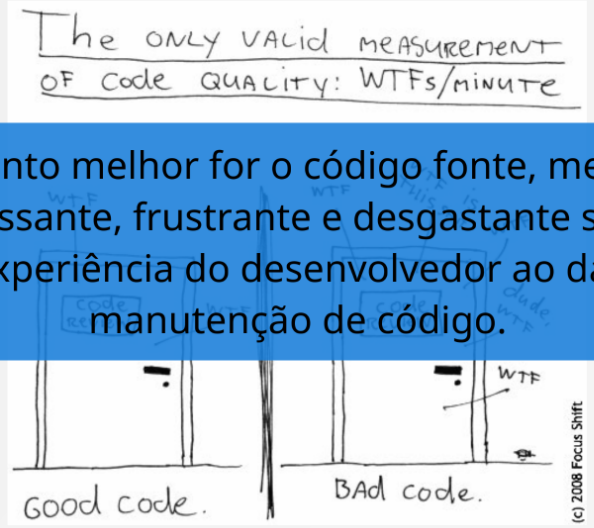
THE
DEVELOPER'S
CONFERENCE

# Como medir a qualidade do meu código?



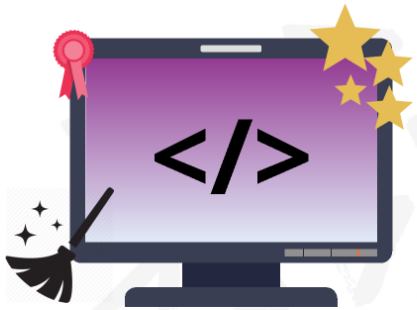

*Robert C. Martin*

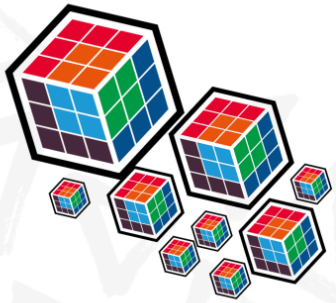
# Como medir a qualidade do meu código?




Quanto melhor for o código fonte, menos estressante, frustrante e desgastante será a experiência do desenvolvedor ao dar manutenção de código.

*Robert C. Martin*







# Codificando funcionalidade e produtividade sem medo

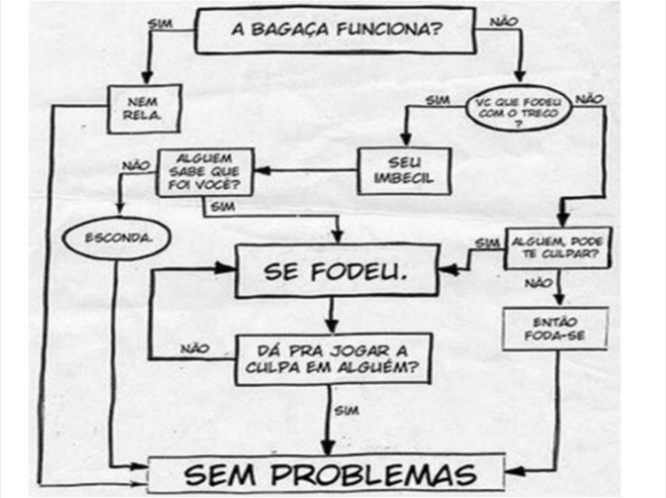





Lorena Dutra
Bacharelado em Sistemas de Informação - UFSM
Mestrado em ciência da Computação - UFSM
Desenvolvedora - Softplan

“ Qualquer um consegue escrever código que um computador entende. Bons programadores escrevem código que humanos entendem”

*- Martin Fowler*




THE DEVELOPER'S CONFERENCE

“ Qualquer um consegue escrever código que um computador entende. Bons programadores escrevem código que humanos entendem”

*- Martin Fowler*

THE DEVELOPER'S CONFERENCE

“ Qualquer um consegue escrever código que um computador entende. Bons programadores escrevem código que humanos entendem”

*- Martin Fowler*

# Obrigada!!

THE DEVELOPER'S CONFERENCE

“ Qualquer um consegue escrever código que um computador entende. Bons programadores escrevem código que humanos entendem”

*- Martin Fowler*

# Obrigada!!

ldutra.info@gmail.com

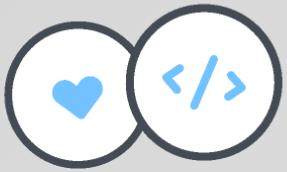
THE DEVELOPER'S CONFERENCE

“ Qualquer um consegue escrever código que um computador entende. Bons programadores escrevem código que humanos entendem”

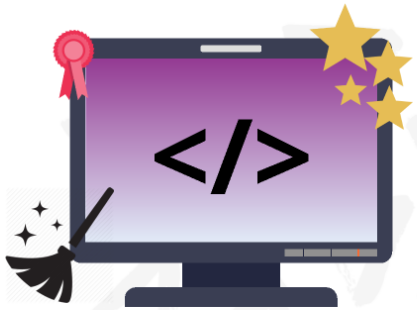
*- Martin Fowler*

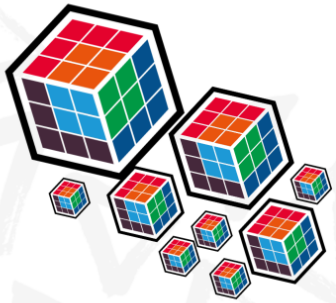
# Obrigada!!

ldutra.info@gmail.com



THE DEVELOPER'S CONFERENCE







# Codificando funcionalidade e produtividade sem medo






Lorena Dutra
Bacharelado em Sistemas de Informação - UFSM
Mestrado em ciência da Computação - UFSM
Desenvolvedora - Softplan

THE
DEVELOPER'S
CONFERENCE