**Guia de Implementação**

SUMÁRIO

[● Introdução 1](#_gjdgxs)

[● Referências 2](#_700jjkgo6aes)

[● Organização e Estilo do Código 2](#_1fob9te)

[● Comentários 2](#_3znysh7)

[● Nomeação 2](#_tyjcwt)

[● Declaração 2](#_3dy6vkm)

[● Reutilização 3](#_1t3h5sf)

[● Tratamento de Erros e de Exceções 3](#_4d34og8)

[● Testes Unitários 3](#_2s8eyo1)

[● Diretrizes Gerais 3](#_17dp8vu)

# Introdução

O Guia de Implementação tem por finalidade fornecer aos desenvolvedores uma fonte para consulta centralizada com relação aos padrões que devem ser seguidos ao criar ou alterar artefatos da implementação. As regras aqui descritas se aplicam a todo o projeto.

Serão descritos a seguir regras para: tamanho e formato de nomes, restrições para número e tamanho de linhas, inclusão de comentários, erros e testes.

# Referências

* Guia - *Oracle Java Code Conventions.pdf*
* *Guidelines for Naming HTML Files and Folders*: https://ed.fnal.gov/lincon/tech\_web\_naming.shtml

# Organização e Estilo do Código

* Tamanho de classe: menor que 500 linhas
* Tamanho de método: menor que 50 linhas
* Tamanho da tabulação: 4 espaços
* Caracteres em uma linha: menos que 150

# Comentários

Comentários devem ser significativos e concisos.

Comentários de uma linha:

// comentário

Comentários de múltiplas linhas:

/\* …

comentário

…\*/

# Nomeação

Os nomes de arquivos devem ser iniciados por letras e podem conter apenas: letras, números, hífens, *underscores* e ponto.

Todos os nomes devem ser tão descritivos quanto possível, sem exceder 40 caracteres.

Arquivos Java devem seguir o padrão PascalCase:

* NomeDaClasse.java
* NomeDaInterface.java

Variáveis devem seguir o padrão CamelCase:

* String nome;
* Long numeroDoRegistro;

Arquivos HTML devem seguir o padrão KebabCase:

* index.html
* cadastro-aluno.html

# Declaração

1. Métodos:

nomeDoMetodo(int arg1, Object arg2, String arg3,

Object arg4) {

...

}

1. Instâncias de classes:

Usuario usuario = new Usuario();

Usuario usuario = new Usuario(

2,

“Nome”,

“email@email.com”,

23,

Grupo.GESTORES);

# Reutilização

Regras gerais para facilitar reutilização:

* Criar as páginas HTML utilizando *fragments*, de acordo com os *templates* já estabelecidos
* Não criar *God Classes*, mas desenvolver as funcionalidades nos arquivos relevantes, tendo em mente a atribuição correta de responsabilidades
* Refatorar o código caso um trecho se repita mais do que duas vezes

# Tratamento de Erros e de Exceções

* Criar uma classe de tratamento de exceções na camada de apresentação.
* Garantir que as exceções subam para a camada de apresentação, usando a cláusula *Throws* nos métodos envolvidos.

# Testes Unitários

Os testes unitários devem cobrir os casos possíveis previstos, utilizando Junit:

* Desenvolver os testes na pasta *test*

# Diretrizes Gerais

Para o desenvolvimento eficaz do projeto, é fundamental estabelecer objetivos e escopo claros, garantindo que todas as partes interessadas compreendam as metas e limitações do projeto. O plano de projeto, incluindo prazos, tarefas e recursos, é essencial para orientar o progresso e a alocação de recursos. A comunicação aberta e a gestão de riscos são cruciais para lidar com desafios e garantir que o projeto permaneça no caminho certo. Priorizar tarefas, promover a colaboração da equipe e integrar testes e controle de qualidade ao longo do ciclo de vida do projeto contribuem para a entrega de produtos ou entregas de alta qualidade. Além disso, a flexibilidade, a documentação adequada e uma avaliação pós-implementação são práticas importantes para o sucesso do projeto, permitindo que a equipe se adapte a mudanças, aprenda com a experiência e continue a melhorar o processo de desenvolvimento.