**Guia de Implementação**

SUMÁRIO

[1. Introdução 2](#_Toc485801465)

[1.1. Referências 2](#_Toc485801466)

[2. Organização e Estilo do Código 2](#_Toc485801467)

[3. Comentários 2](#_Toc485801468)

[4. Nomeação 2](#_Toc485801469)

[5. Declaração 3](#_Toc485801470)

[6. Reutilização 3](#_Toc485801471)

[7. Tratamento de Erros e de Exceções 3](#_Toc485801472)

[8. teste unitário 3](#_Toc485801473)

[9. Diretrizes Gerais 4](#_Toc485801474)

# Introdução

A AGP é um sistema de agenda pessoal que busca ser uma ferramenta que permitirá aos alunos gerenciarem suas matérias e atividades de forma eficiente, com auxílio dos professores. No projeto AGP é desenvolvido com React para o Frontend e Spring para o Backend e portanto as convenções padrões dessas duas técnologias são utilizadas para o desenvolvimento da aplicação.

Este guia tem como objetivo referenciar as convenções utilizadas e mencionar como o projeto deve ser estruturado de acordo com essas convenções.

## Referências

* Guia - Oracle Java Code Conventions.pdf;
* Roldán, C. S. (2023). React 18 Design Patterns and Best Practices: Design, Build, and Deploy Production-ready Web Applications with React by Leveraging Industry-best Practices. Alemanha: Packt Publishing.

# Organização e Estilo do Código

* Arquivos maiores do que 2000 linhas são complicados e devem ser evitados
* Um arquivo consiste em seções que devem ser separadas por linhas em branco e um comentário opcional, identificando cada seção.

# Comentários

Java (multiplas linhas)

/\*

\* Here is a comment.

\*/

Java (unica linha)

/\* comentário. \*/

Java (final de arquivo)

/if (bar > 1) {

//

// // Do a triple-flip.

// ...

//}

//else

// return false;

# Nomeação

Arquivo Java

* NomeClasse.java

Arquivo React / JavaScript

* ArquivoReact.js

# Declaração

1. Padrão de indentação de declaração de métodos em Java:

func() {

if (condition) {

int count; // AVOID!

}

}

1. Padrão de indentação de declaração de classes em Java:

class Sample extends Object {

int ivar1;

int ivar2;

Sample(int i, int j) {

ivar1 = i;

ivar2 = j;

}

int emptyMethod() {}

...

}

# Reutilização

* Criar interfaces entre as camadas arquiteturais: Apresentação, Controle e Dados;
* Aplicar os padrões de atribuição de responsabilidade nas classes (ver padrões GRASP).

# Tratamento de Erros e de Exceções

* Criar uma casse de tratamento de exceções na camada de apresentação.
* Garantir que as exceções subam para a camada de apresentação, usando a clausula Trows nos métodos envolvidos.

Tratamento de exceções em Java:

try {

Scanner s = new Scanner(System.in);

System.out.print("Digite a senha: ");

String senha = s.nextLine();

if(!senha.equals(SENHASECRETA)) {

throw new Exception("Senha invalida!!!");

}

System.out.println("Senha correta!!!\nBem vindo(a)!!!");

} catch (Exception ex) {

System.out.println(ex.getMessage());

}

# teste unitário

# Diretrizes Gerais

Não existem diretrizes e estratégias utilizadas no desenvolvimento do projeto.