**Guia de Implementação**

SUMÁRIO

[1. Introdução 2](#_gjdgxs)

[1.1. Referências 2](#_30j0zll)

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

[3. Comentários 2](#_3znysh7)

[4. Nomeação 2](#_2et92p0)

[5. Declaração 3](#_tyjcwt)

[6. Reutilização 3](#_3dy6vkm)

[7. Tratamento de Erros e de Exceções 3](#_1t3h5sf)

[8. teste unitário 3](#_4d34og8)

[9. Diretrizes Gerais 4](#_2s8eyo1)

# Introdução

Este documento tem por finalidade definir e padronizar as tecnicas, processos e métodos da produção do aplicativo “AgendaAI”. O projeto está fundamentado sobre a linguagem java e o framework springboot. Sendo assim, quaisquer padrões da linguagem e da tecnologia podem e devem ser aplicados durante a manutenção do código. Não somente o código, mas também os “commits” ao github também devem seguir padrões. A lista com toda a convenção usada será colocada na lista de referências, mas os mais importantes são, “docs:” para quaisquer alterações na documentação presente no repositório, “feat:” para qualquer “feature” adicionada, isto é qualquer alteração que adicione uma funcionalidade a aplicação como um todo ou ao código, “fix:” para qualquer alteração que apenas remova um bug ou conserte determinado trecho do código. Como o aplicativo não é um monolith e sim uma serie de microserviços que se integram, cada serviço deve ter apenas código que diga a respeito delas mesmo, a integração das API’s não é responsabilidade delas. A aplicação deve ser implementada no server-side.

## Referências

* Guia - https://www.oracle.com/java/technologies/javase/codeconventions-contents.html;
* Guia - https://www.conventionalcommits.org/pt-br/v1.0.0-beta.4/

# Organização e Estilo do Código

* Métodos podem ou não ter um comentário acima dele explicando sua função, e fica a critério do desenvolvedor julgar a complexidade do código para documentá-lo ou não via comentários.
* Arquivos devem estar, se possível, com menos que 3000 linhas de código.
* Métodos ou funções devem evitar, sempre que possivel, estar acima de 100 linhas de código.

# Comentários

Comentários podem ser em uma ou mais linhas, se atentando ao uso de cada linguagem.

* Múltiplas linhas (java):  
  /\*

\* Exemplo de comentario de multiplas linhas em java.

\* /

* Única linha (java):

// Exemplo de comentario em unica linha

* Unica linha (TypeScript):

// Exemplo de comentario em unica linha typescript.

ou

/\*\* Comentario \*/

* Comentario em multiplas linhas (TypeScript):

/\*\*

\*

\*/

* Comentarios em html:  
  <!-- Comentario em html–>

# Nomeação

Arquivo Java

* NomeClasse.java

Arquivos Typescript (Angular)

* nomeDoComponent.component.TipoDeArquivo

# Declaração

Métodos devem seguir o padrão camel case.

1. Padrão de indentação de declaração de métodos:

someMethod(int anArg, Object anotherArg, String yetAnotherArg,

Object andStillAnother) {

...

}

1. Se possível em angular, a função, método ou variável deve ser tipado.

# Reutilização

* Todo código do submodulo AgendaAi-API deve ser dividido nos padrões do SpringBoot.
* *Todo codigo do submodulo AgendaAi-APP deve ser dividido nos padrões do Angular com componentes.*

# Tratamento de Erros e de Exceções

* Se o retorno for um array ou json, retorna uma lista vazia, e não um erro.
* No aplicativo, as interfaces ou classes responsáveis por armazenar a informação devem se atentar aos possíveis retornos null e implementar coalescencia nula.

# teste unitário

Não são utilizados testes de JUNIT, qualquer teste deve ser implementado com o github actions.

# Diretrizes Gerais

O arquivo README.MD do projeto deve estar sempre atualizado com instruções claras de como rodar o servidor e com as principais informações. Novos commits devem estar em branchs seguindo o conventional commits, também chamado de padrão de commits do Angular.