**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 3](#_Toc485801474)

# Introdução

Este documento visa padronizar a organização e estilização do código a ser desenvolvido

## Referências

<https://devluciano.wordpress.com/tag/padroes-de-nomenclaturas-c/>

# Organização e Estilo do Código

* Tamanho de classe: não mais que 1000 linhas;
* Tamanho de método: não mais que 50 linhas;
* Padrão de nomeação de classes: PascalCase;
* Padrão de nomeação de métodos: camelCase;
* Padrão de nomeação de variáveis: camelCase;
* Padrão de nomeação de pacotes: camelCase;

# Comentários

Padrão para comentarios de Classes

/\*\*

\* <Descrição da funcionalidade da Classe>

\*

\*/

Padrão para comentarios de Métodos

// <Descrição simplificada da funcionalidade do método>

# Nomeação

Regras Gerais:

* Não utilizar acentos.
* Não utilizar espaços.

Pacotes

* controller
* model
* userInterface

**Model**

Entidades

* NomeClasse.cs

DAO

* NomeClasseDAO.cs

**Controller**

* NomeClasseCtrl.cs

**View**

User Interface

* nomePaginaView.cs

Variáveis

* nomeVariavel

Scripts SQL

* NOMEDOSCRIPT.sql

Enumeradores

* enumNomeEnum.cs
* Exemplo: enumFuncao.cs

Componentes

* TextBox
  + txt
  + Exemplo: txtNome
* Label
  + lbl
  + Exemplo: lblNome
* ComboBox
  + cmb
  + Exemplo: cmbTipos
* PictureBox
  + pib
  + Exemplo: pibImagens
* DataGridView
  + dgv
  + Exemplo: dgvFuncionarios
* Radio Button
  + rbt
  + Exemplo: rbtOpcaoPagamento
* CheckBox
  + chk
  + Exemplo: chkPago
* ListBox
  + lst
  + Exemplo: lstEtapas
* Button
  + btn
  + Exemplo: btnCadastrar

# Declaração

* Instâncias de classes devem ser declaradas com prefixo “obj”
  + objObra
* Declarar uma variável por linha

# Reutilização

* Aplicar os padrões de atribuição de responsabilidade nas classes.

# Tratamento de Erros e de Exceções

* Garantir que as exceções subam para a camada de apresentação, utilizando-se do TryCatch.

# teste unitário

* Elaborar roteiro de testes com NUnit.

# Diretrizes Gerais

Não se aplica.