
Desenvolvimento Baseado em Padrões e Framework

Framework

Um **framework** é uma estrutura de programação que fornece um conjunto de ferramentas, bibliotecas, padrões e diretrizes para facilitar o **desenvolvimento de software**. Ele define uma arquitetura básica para um sistema, com módulos e componentes interconectados, além de oferecer uma base para a implementação de funcionalidades específicas.

Os **frameworks** são úteis porque permitem que os desenvolvedores se concentrem na lógica de negócios e na funcionalidade específica de seu **software**, em vez de perder tempo construindo estruturas básicas. Isso pode economizar tempo e esforço no desenvolvimento e resultar em um código mais organizado, legível e escalável.

Eles podem incluir bibliotecas de código, ferramentas de compilação e depuração, estruturas de arquitetura de software, modelos de design e muito mais.

Os **frameworks** são usados para acelerar o processo de desenvolvimento, permitindo que os desenvolvedores se concentrem em problemas específicos de negócios ou de aplicativos em vez de ter que escrever código básico repetitivo do zero. Eles também ajudam a padronizar o código e a manter a consistência entre os desenvolvedores e as equipes de desenvolvimento.

❖ Vantagens em Programar Utilizando Framework

Eficiência

- Processo de desenvolvimento facilitado
- Redução de linhas de códigos
- Códigos mais limpos
- Bibliotecas e funcionalidades automatizadas
- Reutilização de código
- Poupa tempo e esforço da equipe de desenvolvimento

Custo de Desenvolvimento

- Distribuição gratuita open source
- Poupa horas de trabalho diminuindo o custo do projeto
- Funcionalidades bem definidas minimizando o retrabalho
- Gestão do conhecimento

Suporte Técnico

- Documentação oficial
- Comunidade de desenvolvedores
- Cursos específicos
- Profissionais especializados

Confiabilidade

- Largamente testado e utilizado
- Vive em constante evolução
- Rápida correção de falhas junto à comunidade
- Desempenho e segurança
- Desenvolvimento baseado em padrões

❖ *Tipos de Frameworks para Desenvolvimento de Software*

Frameworks de Front-End

- Trabalha efetivamente junto ao layout de um software
- Exemplos de Framework Font-End:
- Primefaces
- Bootstrap
- jQuery
- Material-UI
- Dentre vários outros...

Frameworks de Back-End

- Trabalha efetivamente junto a camada de regras de negócio de um software, basicamente com quem o layout se comunica.
- Exemplos de Framework Back-End:
- Spring Framework
- Grails
- JavaServer Faces
- Struts
- Dentre vários outros...

Frameworks de Persistência

- Trabalha efetivamente com a construção de estrutura de armazenamento em banco de dados.
- Exemplos de Framework de persistência:
- Hiberante
- Spring Data
- Eclipse Link
- Torque
- Dentre vários outros...

❖ O que é Importante na Escolha de um Framework?

Tempo de Mercado

- Minimiza a possibilidade de bug
- Ganho de performance
- Consistência
- Durabilidade
- Segurança
- Confiável

Sólida Comunidade de Desenvolvedores

- Fórum de discussão confiável
- Suporte para correção de possíveis falhas
- Troca de experiência
- Consulta de soluções já implementadas

Documentação de Qualidade

- Informação técnica do funcionamento do framework
- Configuração para execução
- Padrão de trabalho
- Exemplos práticos
- Orientações diretas
- Vários idiomas
- Uma documentação de qualidade diminui a curva de aprendizado

Popularidade do Framework

- Oportunidade de trabalho
- Facilidade na contratação de desenvolvedores
- Comunidade fortalecida
- Consequentemente vários cursos sobre a tecnologia

Protótipo Para Teste de Implementação

- Desenvolver algumas funcionalidades para simular o real uso do framework
- Definir padrão arquitetural da aplicação
- Familiarizar com os padrões de desenvolvimento do framework
- Efetuar testes de carga para avaliar a performance da ferramenta
- Dentre outros...

❖ *Aplicações WEB*

- Inicialmente a camada de Layout de uma aplicação **WEB** pode ser composta de **HTML**, **CSS** e **JAVA SCRIPT**.
- **HTML** ---> Linguagem de marcação e hipertexto.
- **CSS** ---> Folha de estilo em cascata.
- **JAVA SCRIPT** ---> Linguagem de programação.

Layout e Aplicações WEB

Os layouts **WEB** são compostos por componentes como:

- Botões
- Campos de texto
- Hiperlinks
- Tabelas
- Listagens
- Etc...
- Segue abaixo um exemplo de uma aplicação **WEB**

🔗 Formulário para inscrição em um torneio de futebol



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/unipar/'. The page title is 'Inscrição para torneio de futebol'. The form contains four input fields: 'Nome do atleta', 'RG:', 'Data de nascimento:', and 'Nome do seu time:'. There is a button labeled 'Enviar dados'.

- O formulário é composto por alguns componentes **WEB**, porém não possui estilo (formatação) e está completamente estático.

🔗 Como podemos melhorar isso? ---> **R:** Com **CSS** e **JAVASCRIPT**

❖ *Código Fonte do Estilo Aplicado (Sem Framework)*

- Grande volume de código
- Difícil manutenção
- Dificuldade em padronização
- Possíveis falhas

```

<html>
  <head>
    <title>Inscrição para torneio de futebol</title>
  </head>
  <body style="background-color: #CEECF5;">
    <h1 style="font-family: Verdana; border-bottom: solid 1px;">Inscrição para torneio de futebol</h1>

    <span style="font-family: Verdana; display: block; margin-top: 10px;">Nome do atleta:</span>
    <input type="text" style="display: block; width: 400px;">

    <span style="font-family: Verdana; display: block; margin-top: 10px;">RG:</span>
    <input type="text" style="display: block;">

    <span style="font-family: Verdana; display: block; margin-top: 10px;">Data de nascimento:</span>
    <input type="text" placeholder=" / / " style="display: block; width: 100px;">


    <span style="font-family: Verdana; display: block; margin-top: 10px;">Nome do seu time:</span>
    <input type="text" style="display: block; width: 350px;">

    <input type="submit" value="Enviar dados" style="background-color: #58B2FA; border: none; border-radius: 15px; color: #FFFFFF; margin-top: 10px; font-weight: bold; padding: 10px;">

  </body>
</html>

```

Destaque para os códigos de estilo



❖ *Código Fonte do Estilo Aplicado (Com Framework)*

- Baixo volume de código
- Facilita a manutenção
- Código padronizado
- Reaproveitamento de código
- Menor chance de falhas

```

<html>
  <head>
    <title>Inscrição para torneio de futebol</title>
  </head>
  <body class="background">
    <h1 class="title">Inscrição para torneio de futebol</h1>

    <span class="label">Nome do atleta:</span>
    <input type="text" class="fieldText400"/>

    <span class="label">RG:</span>
    <input type="text" class="fieldText10"/>

    <span class="label">Data de nascimento:</span>
    <input type="text" placeholder=" / / " class="fieldText100"/>

    <span class="label">Nome do seu time:</span>
    <input type="text" class="fieldText350"/>

    <input type="submit" value="Enviar dados" class="button"/>
  </body>
</html>

```

❖ *HTML (Hyper Text Markup Language)*

- É uma linguagem de marcação
- Utilizada como estrutura na criação de páginas para **WEB**
- É interpretado por Navegadores **WEB**

↳ **Firefox, Google Chrome, Opera, etc...**

- Seus arquivos possuem extensão **.html**

↳ **Ex: index.html**

Um documento **HTML** é formado por **TAGs**:

- A tag **<HTML>** indica o início da página **html**
- **<HEAD>** Seu conteúdo não será exibido no navegador **WEB**, pois dentro dessa tag contém as configurações da página.
- **<BODY>** é o corpo da página, representa o conteúdo **HTML** que será exibido no navegador **WEB**.

```

<html>
  <head>
  </head>
  <body>
  </body>
</html>

```

INICIO DO DOCUMENTO HTML

CABEÇALHO DA PÁGINA

CORPO DA PÁGINA

❖ CSS (Cascading Style Sheets)

É um mecanismo para adicionar estilo (formatação) em documentos **HTML**.

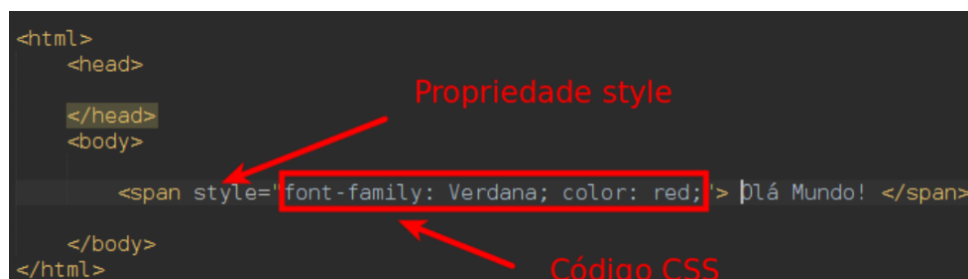
O **CSS** tem o poder de alterar cores, fontes, espaçamentos, alinhamentos, etc...
Em documentos **HTML**.

O código **CSS** pode ser aplicado de 3 maneiras:

- Diretamente na **TAG HTML** utilizando a propriedade **style**.
- Dentro do **HEAD** em um documento **HTML** utilizando a **TAG <style>**.
- Importando um arquivo com extensão **.css** em um documento **HTML**.

Diretamente na **TAG HTML** utilizando a propriedade **style**:

- A propriedade **style** é encontrada nas **TAGs** que compõem o corpo do documento **HTML**.

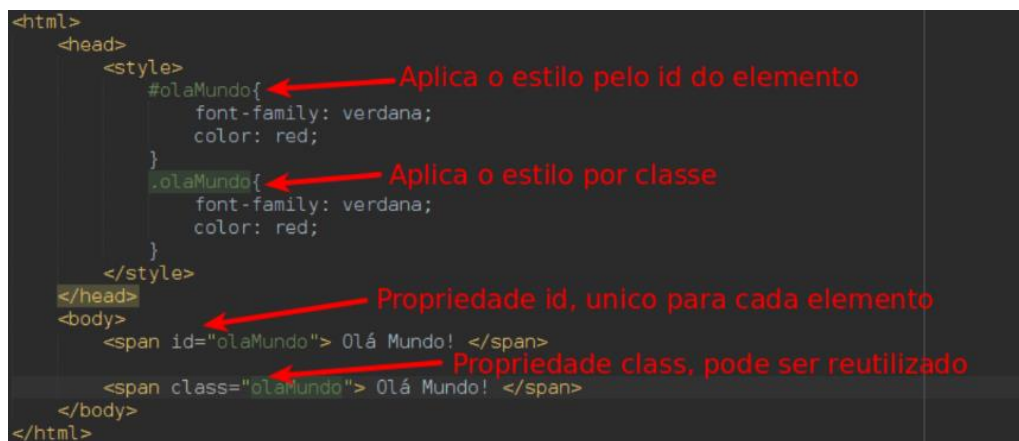


```
<html>
<head>
</head>
<body>
  <span style="font-family: Verdana; color: red;"> Olá Mundo! </span>
</body>
</html>
```

Diagram illustrating inline CSS application. A red box highlights the `style="font-family: Verdana; color: red;"` attribute in the `` tag. A red arrow points to this box with the label "Propriedade style". Another red arrow points to the text `font-family: Verdana; color: red;` with the label "Código CSS".

Dentro do **HEAD** em um documento **HTML** utilizando a **TAG <style>**:

- Se o estilo for aplicado pelo **ID** do elemento, não pode ser reutilizado, pois no **HTML** cada elemento deve possuir um **ID** único na página.
- Se o estilo for aplicado por **class** (classe), pode ser reutilizado para outras **TAGs** no documento **HTML**, obtendo o reaproveitamento de código.

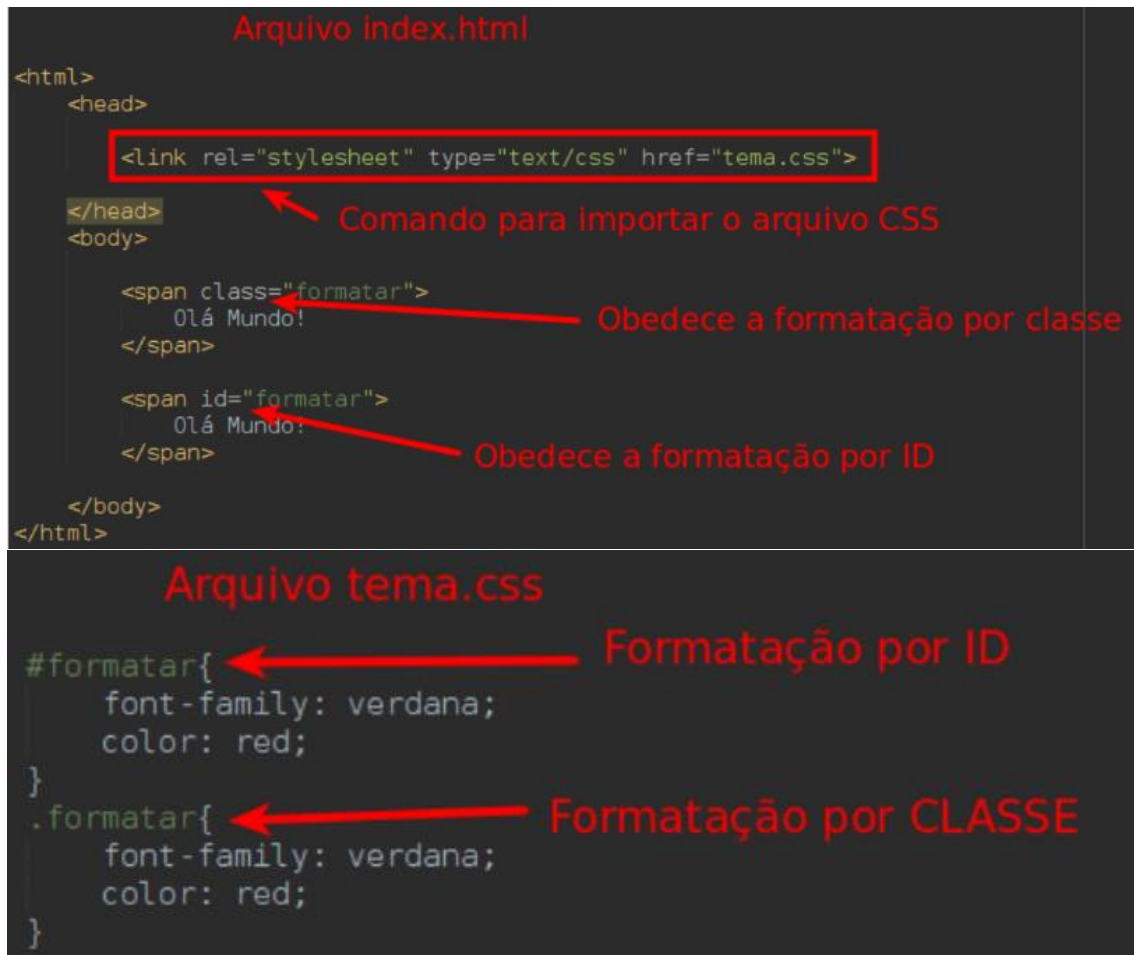


```
<html>
<head>
  <style>
    #olaMundo{
      font-family: verdana;
      color: red;
    }
    .olaMundo{
      font-family: verdana;
      color: red;
    }
  </style>
</head>
<body>
  <span id="olaMundo"> Olá Mundo! </span>
  <span class="olaMundo"> Olá Mundo! </span>
</body>
</html>
```

Diagram illustrating CSS application via the `<style>` tag in the head. Red arrows point to the `#olaMundo{}` and `.olaMundo{}` selectors with labels "Aplica o estilo pelo id do elemento" and "Aplica o estilo por classe" respectively. Another red arrow points to the `id="olaMundo"` attribute with the label "Propriedade id, unico para cada elemento". A final red arrow points to the `class="olaMundo"` attribute with the label "Propriedade class, pode ser reutilizado".

Importando um arquivo com extensão **.CSS** em um documento **HTML**.

- Com a **tag <link>** é possível importar arquivos **.CSS** com estilos pré-definidos.
- Promovendo uma grande reutilização de **código**
- Esse tipo mecanismo é considerado como um **Framework**



JavaScript

JavaScript é uma linguagem de programação de alto nível que é frequentemente utilizada para criar conteúdo dinâmico em páginas da **web**.

É uma linguagem de script, o que significa que é executada no navegador do usuário em tempo real, em vez de precisar ser compilada antes da execução.

JavaScript é uma das três tecnologias fundamentais da **web**, junto com **HTML** e **CSS**, e é amplamente utilizado em desenvolvimento **web**, **jogos**, **aplicativos móveis** e outras aplicações.

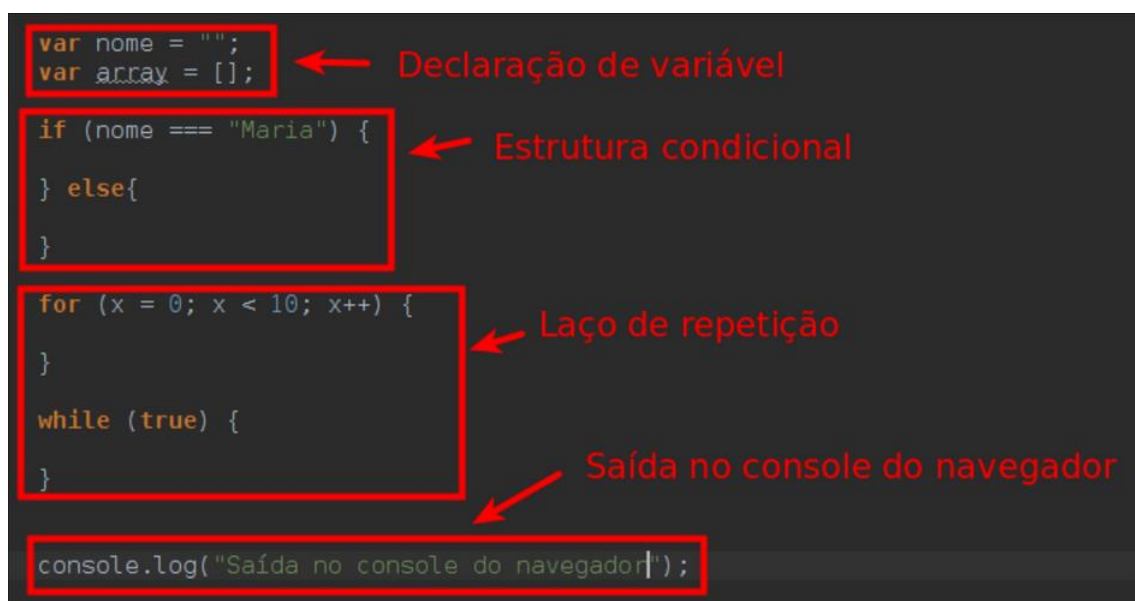
JavaScript permite que os desenvolvedores adicionem recursos interativos às suas páginas da web, como botões de rolagem, menus suspensos, elementos de formulário dinâmicos e animações.

Ele também é usado para criar aplicativos da **web** em grande escala, como aplicativos de gerenciamento de projetos, plataformas de comércio eletrônico e ferramentas de produtividade.

Como uma linguagem de programação versátil e amplamente utilizada, o **JavaScript** tem uma grande comunidade de desenvolvedores que continuam a expandir e melhorar suas capacidades, **podendo ser utilizado tanto no front-end como no backend**.

❖ *Sintaxe da Linguagem*

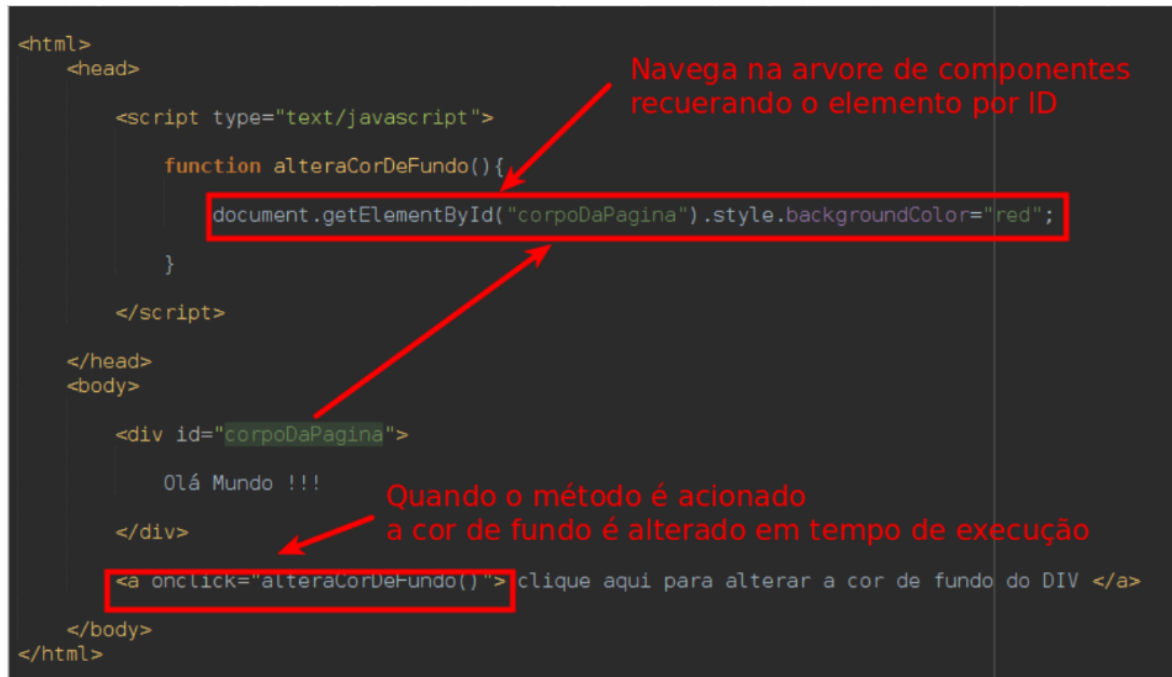
O **JavaScript** possui uma sintaxe bem intuitiva, segue abaixo alguns exemplos:



❖ *DOM (Document Object Model)*

- O **DOM** possui métodos para acesso aos componentes em um documento **HTML**
- Permite a alteração do comportamento dos elementos
- Podemos criar, remover e alterar propriedades dos elementos
- Possui um papel importante no desenvolvimento **WEB**

O exemplo abaixo demonstrar a alteração da cor de fundo do elemento **<div>** vincula ao evento **onclick** de um botão:



❖ *Os componentes HTML possuem vários eventos:*

- **OnClick:** Ao clicar com o mouse
- **Onkeyup:** Ao pressionar a tecla no teclado
- **Onmouseout:** Ao passar o mouse
- **Onchange:** Ao alterar opções (para **combobox**)
- **Onfocus:** Ao adicionar o foco em uma **input**
- **Onload:** Ao carregar um componente

❖ *Exemplo de Implementação*

- O **JavaScript** pode ser implementado:
- Entre a **TAG <script>** no **<head>** da página **HTML**
- Diretamente nos eventos dos componentes **HTML**
- Em um arquivo separado e importado dentro de uma página **HTML**

Entre a **TAG <script>** no **<head>** da página **HTML**

```
<html>
  <head>
    <script type="text/javascript">
      alert("Olá Mundo!");
    </script>
  </head>
  <body>

  </body>
</html>
```

Boloco de implementação do JavaScript

Diretamente nos eventos dos componentes **HTML**

```
<html>
  <head>

  </head>
  <body>
    <a onclick="alert('Olá Mundo')"> Clique aqui </a>
  </body>
</html>
```

Implementação no evento onclick

Em um arquivo separado e importado em um arquivo **HTML**

Arquivo javascript.js

```
function testeImplementacao() {
  alert("Olá Mundo");
}
```

Arquivo index.html

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <a onclick="testeImplementacao()"> clique aqui </a>
  </body>
</html>
```

Código para importar o arquivo javascript.js

O método "testeImplementacao()" será invocado no evento onclick

❖ *Considerações Finais*

JavaScript:

- Aumenta a interatividade em uma página WEB
- Utilizando o DOM podemos manipular os elementos HTML
- Da a possibilidade de criar bibliotecas para automatização de processos.
- Possui recursos para reaproveitamento de código
- Diversos frameworks WEB do mercado são baseados em JavaScript
- Extremamente importante para o desenvolvimento de aplicações WEB

Framework:

- Não reinventar a roda
 - Produtividade
 - Agilidade
 - Reaproveitamento de código
 - Código consistente e padronizado
 - Software de qualidade
-
- Existem possibilidades de criar o próprio framework
 - Centralização de funções
 - Reaproveitamento de código fonte
 - Economia de tempo do time
 - Padronização de implementação
 - Código limpo com fácil manutenção