Universidade Tecnológica Federal do Paraná  
Engenharia de Software - Curso de Arquitetura de Software (AS27S)

INSTRUTOR: Prof. Dr. Gustavo Santos

Kamila Antunes de Souza Neves, 2204690

linha horizontal

# CCH - Design Patterns Builder – Spider-Man

## Problema

O problema abordado neste código é a criação de objetos do Spider-Man com diferentes características opcionais, como a cor da roupa, a presença do disparador de teias e a força. O objetivo é criar uma estrutura flexível que permita a construção desses objetos passo a passo, separando a lógica de construção da representação final do objeto.

## Descrição da Solução

A solução adotada neste código é a implementação do padrão de projeto criacionais do tipo Builder. O padrão Builder separa a construção de um objeto complexo de sua representação final, permitindo a criação de objetos com diferentes configurações opcionais.

O objetivo deste código é demonstrar a aplicação do padrão de projeto criacionais do tipo Builder para criar objetos do Spider-Man na linguagem de programação JavaScript.

Possuindo duas principais classes:

* **SpiderManBuilder**: classe que implementa o padrão Builder. Ela tem métodos para definir as características opcionais do Spider-Man, como a cor a roupa, o disparador de teia e a sua força. Cada método retorna a própria instância do **SpiderManBuilder**, permitindo encadear as chamadas dos métodos.
* **SpiderMan**: classe que representa o objeto Spider-Man. Possui as propriedades do Spider-Man, como a cor da roupa, se possui disparador de teia e se possui força. Ainda, possui métodos para exibir informações sobre o Spider-Man, disparador de teia e a força.

## Visão Geral [exemplo]

Diagrama

Descrição gerada automaticamente

* A classe **SpiderMan** representa o objeto Spider-Man e possui as propriedades **suitColor**, **webShooters** e **superStrength**. Ela também contém os métodos **useWebShooters()**, **useSuperStrength** e **displayInfo()** para realizar as ações relacionadas ao Spider-Man.
* A classe **SpierManBuilder** é responsável pela construção do objeto **Spider-Man**. Ela possui uma propriedade **spiderMan** que representa o objeto sendo construído. Os métodos **setSuitColor()**, **setWebShooters()**, **setSuperStrength()** permitem definir características opcionais do Spider-Man passo a passo. O método **build()** retorna o objeto Spider-Man final.

## Exemplo de Código JavaScript

O código pode ser acessado através do repositório disponibilizado através do link – [CCH 1 Padrões de projetos criacionais](https://github.com/kamilaantunes/engenharia_de_software/tree/main/7%C2%B0%20per%C3%ADodo/Arquitetura%20de%20software/Padr%C3%B5esDeProjetosCriacionais).

## Consequências [vantagens e desvantagens]

* Vantagens
  + Flexibilidade: o padrão Builder permite criar objetos com diferentes configurações opcionais, oferecendo flexibilidade para ajustar as características do Spider-Man;
  + Separação de responsabilidades: a separação entre o **SpiderManBuilder** e a classe **SpiderMan** permite uma divisão clara entre a lógica de construção e a representação do objeto, facilitando a manutenção e entendimento do código;
  + Encadeamento de métodos: a capacidade de métodos do **SpiderManBuilder** torna o código mais legível e expressivo, permitindo a construção passo a passo intuitiva.
* Desvantagens
  + Complexidade adicional: a implementação do padrão Builder pode adicionar alguma complexidade ao código, com a necessidade de criar uma classe separada para atuar como o construtor e a necessidade de encadear os métodos corretamente.
  + Overhead: o uso do padrão Builder pode introduzir algum overhead, pois se faz necessário criar uma classe adicional e invocar vários métodos para construir o objeto Spider-Man.