Frameworks

Ian Akira Ganico

Sorocaba

24 de Agosto de 2020

**Introdução**

Usar frameworks é algo que já faz parte do dia a dia da maioria dos desenvolvedores, especialmente de quem trabalha com um grande número de projetos que usam funções similares. Afinal, a possibilidade de reutilizar códigos com poucas alterações ajuda a poupar tempo.

Isso porque o framework dá uma caixa de ferramentas para o programador, que vai além do que é oferecido pela linguagem. Seu conceito, porém, pode ser confuso em relação a outras formas de aproveitar códigos em vários projetos, como a orientação a objetos e às classes.

A grande diferença é que o método opera de forma muito mais profunda, com vantagens e desvantagens. Por isso, neste post, explicaremos o que é framework e qual é seu impacto no desenvolvimento de projetos.

**Definição**

Um framework em [desenvolvimento de software](https://pt.wikipedia.org/wiki/Desenvolvimento_de_software), é uma [abstração](https://pt.wikipedia.org/wiki/Abstra%C3%A7%C3%A3o) que une [códigos](https://pt.wikipedia.org/wiki/C%C3%B3digo) comuns entre vários projetos de software provendo uma funcionalidade genérica. Um *framework* pode atingir uma funcionalidade específica, por configuração, durante a programação de uma aplicação. Ao contrário das [bibliotecas](https://pt.wikipedia.org/wiki/Biblioteca_(computa%C3%A7%C3%A3o)), é o *framework* quem dita o fluxo de controle da aplicação, chamado de [Inversão de Controle](https://pt.wikipedia.org/wiki/Invers%C3%A3o_de_Controle).

Um *Framework,* ou arcabouço conceitual, é um conjunto de conceitos usado para resolver um problema de um domínio específico.

***Framework* conceitual** não se trata de um software executável, mas sim de um [modelo de dados](https://pt.wikipedia.org/wiki/Modelo_de_dados) para um domínio.

***Framework* de** [**software**](https://pt.wikipedia.org/wiki/Software) compreende de um conjunto de **classes** implementadas em uma [linguagem de programação](https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o) específica, usadas para auxiliar o desenvolvimento de software.

O *framework* atua onde há funcionalidades em comum a várias aplicações, porém para isso as aplicações devem ter algo razoavelmente grande em comum para que o mesmo possa ser utilizado em várias aplicações.

[Padrões de projeto de software](https://pt.wikipedia.org/wiki/Padr%C3%B5es_de_projeto_de_software) não se confundem com *frameworks*, pois padrões possuem um nível maior de abstração. Um *framework* inclui código, diferentemente de um padrão de projeto. Um *framework* pode ser modelado com vários padrões de projeto, e sempre possuem um domínio de uma aplicação particular, algo que não ocorre nos padrões de projeto de software.

Frameworks possuem vantagens, tais como: maior facilidade para a detecção de erros, por serem peças mais concisas de software; concentração na abstração de soluções do problema que estamos tratando; eficiência na resolução dos problemas e otimização de recursos.

As funções do framework têm uma grande variedade de parâmetros, garantindo ao desenvolvedor a possibilidade de fazer personalizações, de acordo com as necessidades do projeto. Para isso, são usados princípios de orientação a objeto, como a abstração, o polimorfismo e a herança.

Aliás, vale ressaltar que a integração entre as diferentes funções dessa ferramenta é uma de suas principais características. Isso significa que as ferramentas são feitas de forma aberta, para que se adaptem a uma grande quantidade de situações. Os pontos providos pelo framework são chamados de frozen spots ou hook points. Já a instanciação e a personalização criadas pelo desenvolvedor são denominadas de host spots.

**Diferencial**

A criação de funções e a modularização também existem fora dos frameworks, o que pode causar certa confusão. Bibliotecas de classes, por exemplo, são uma implementação em que as funções podem ser importadas para uso em diversos projetos. A grande diferença do framework é a integração entre suas diferentes funções. Enquanto em bibliotecas de classes as funções operam de forma relativamente independente, em um framework há relações já embutidas de dependência entre os componentes.

Outro diferencial é a forma como o programa flui. No caso das bibliotecas, por exemplo, é o programa que controla o fluxo e faz as importações. Já quando se usa um framework, é ele quem acessa as funções criadas pelo usuário.

**Frameworks em orientação a objeto**

Especificamente em [orientação a objetos](https://pt.wikipedia.org/wiki/Orienta%C3%A7%C3%A3o_a_objetos), *framework* é um conjunto de [classes](https://pt.wikipedia.org/wiki/Classe_(programa%C3%A7%C3%A3o)) com objetivo de reutilização de [arquitetura de software](https://pt.wikipedia.org/wiki/Arquitetura_de_software), provendo um guia para uma solução em um domínio específico de *software*. *Framework* se diferencia de uma simples biblioteca, pois esta se concentra apenas em oferecer implementação de funcionalidades, sem definir a reutilização de uma solução de arquitetura.

Muitos engenheiros acreditam que a arquitetura é determinada pelos requisitos e por isso esperam que a fase de [engenharia de requisitos](https://pt.wikipedia.org/wiki/Engenharia_de_requisitos) esteja finalizada para então iniciar sua. Porém, apenas uma fração dos requisitos específicos do sistema têm influência na arquitetura. A identificação dos requisitos que são significantes para a arquitetura pode ser respondida através de um ***framework* conceitual** desenvolvido especialmente para um domínio específico, uma vez que esta resposta é muito dependente do domínio. Avançar para a fase de projeto ou mesmo iniciar a implementação do sistema não quer dizer que a definição da arquitetura esteja finalizada. Isto significa que o detalhamento obtido até então já é suficiente para prosseguir com o projeto de uma parte do sistema.

**Frameworks em empresas**

Em administração, um *framework* é uma estrutura conceitual básica que permite o manuseio homogêneo de diferentes objetos de negócio. Serve para incrementar a disciplina de gestão e predefinir entregáveis comuns para cada objeto de negócio.

Pode ser visto também como uma tática bem definida para manipular com destreza ambientes organizacionais complexos. Um framework deve prover sugestões de solução para uma família de problemas semelhantes.

Exemplos de frameworks para gestão:

* [ISO 9000](https://pt.wikipedia.org/wiki/ISO_9000)
* [ISO 14000](https://pt.wikipedia.org/wiki/ISO_14000)
* [OHSAS 18000](https://pt.wikipedia.org/w/index.php?title=OHSAS_18000&action=edit&redlink=1)
* [ITIL](https://pt.wikipedia.org/wiki/ITIL)
* [COBIT](https://pt.wikipedia.org/wiki/COBIT)
* [CMM](https://pt.wikipedia.org/wiki/CMM)
* [HACCP](https://pt.wikipedia.org/wiki/HACCP)
* [SCRUM](https://pt.wikipedia.org/wiki/SCRUM).

**Conclusão**

É possível encontrar diversas definições para **Framework**, algumas simples, outras mais elaboradas, mas o ponto comum entre todas é a **reusabilidade**.

Assim um Framework tem como principal objetivo resolver problemas recorrentes com uma abordagem genérica, permitindo ao desenvolvedor focar seus esforços na resolução do problema em si, e não ficar reescrevendo software. Então pode-se dizer que é um conjunto de bibliotecas ou componentes que são usados para criar uma base onde sua aplicação será construída para a resolução do problema.

**Referências**

Framework. Wikipedia, a enciclopédia livre, Acesso em 24 de Agosto de 2020.

Disponível em: < <https://pt.wikipedia.org/wiki/Framework>>.

Entenda oque é um Framework. Gaea, Acesso em 24 de Agosto de 2020.

Disponível em: < https://gaea.com.br/entenda-o-que-e-framework/>.