# O Pensamento Computacional

Um bom pensamento computacional consiste na formulação e resolução de problemas juntamente com um pensamento sistemático e eficiente. É, portanto, uma habilidade generalista que pode ser aplicada em diversos campos como: escrita, matemática etc.

- É baseado nos pilares de:
- Decomposição;
- Reconhecimento de padrões;
- Abstração;
- Design de algoritmos;

A decomposição pode ser entendida como uma metáfora na receita de um bolo, pois é relacionado a decomposição e segmentação de um todo em etapas para atingir o mesmo fim. Ademais, pode ser sequencial e paralelo.

Ao criar um app/site é preciso decompor:

1. Sua finalidade;
2. Interface;
3. Funcionalidades;
4. Pré-requisitos.

O reconhecimento de padrões pode ser definido pela similaridade e diferenças entre os problemas e reconhecer suas tendências. Um exemplo de padrão comum são as bolsas de valores porque quando cai, os investidores vendem e quando sobe, muitos compram (efeito sardinha). Nesse processo, a análise de comparação é fortemente atrelada.

A abstração consiste na abstração de algo particular (como software) e generalizar essa abstração (como as classes). Já o Design de algoritmos define passo a passo a resolução do problema de uma forma automatizada e organizada.

Exemplos na computação de abstração:

- Em algoritmo: input, operação, output...
- Estrutura de dados: árvore, grafos, listas...
- Linguagens de programação: Java, C++ etc.

Além dos pilares existe um processo contínuo extremamente importante que é o ciclo de analisar, testar e refinar um processo. O pensamento

computacional depende do pensamento humano, pois, apenas este é capaz de fazer abstrações e automatizá-las.

# Habilidades Complementares

Se fazem dias presentes:

- Raciocínio lógico;
- Refinamento.

O raciocínio lógico é classificado em três vertentes: a indução — observar acontecimentos reais e extrapolar eles para o campo da ideia e determinar leis e teorias. A dedução — realizar previsões e explicações baseados naquela lei estabelecida pela indução. A abdução — muito utilizado em diagnósticos e áreas investigativas, pois aposta na opção mais lógica, porém não deixa de ser errônea.

Logo um bom desenvolvedor deve ser capaz de:

- Encontrar soluções eficientes;
- Otimizar processos;
- Simplificar linhas de código;
- Fazer funções bem definidas.

Na linguagem de algoritmos é necessário:

- Compreender o problema;
- Definição de dados de entrada;
- Definir o processamento;
- Definir dados de saída;
- Utilizar métodos de construção (narrativa, fluxograma ou pseudocódigo);
- Teste e diagnóstico (refinamento).

Benefícios: Melhor uso dos recursos, fácil manutenção e melhorar o código e o algoritmo implementado.