Pensamento computacional é o processo de pensamento envolvido na expressão de soluções em passos computacionais ou algoritmos que podem ser implementados no computador. É uma habilidade generalista, não disciplina acadêmica. Baseia-se em 4 pilares:

-Decomposição (divisão de um problema complexo em problemas menores resolvíveis):

A estratégia desse pilar é quebrar e determinar partes menores gerenciáveis. Após essa quebra e análise segmentada, voltamos a combinar os elementos para recompor o problema original. A análise pós-decomposição pode ser sequencial (onde as tarefas são correlacionadas e seguem uma fila/ordem) ou paralela (mais eficiente e rápida que a sequencial, vez que diversas tarefas podem ocorrer ao mesmo tempo em paralelo, sem serem codependentes). Exemplo: Usamos a decomposição ao cozinhar. Pensamos primeiro no prato inteiro, concluído. Porém, em seguida, temos que pensar em cada ingrediente que será usado, e utilizá-los de forma sequencial para que o resultado se mostre no final das tarefas menores. Para criar um app, primeiro temos que pensar em sua finalidade, interface, funcionalidades, pré-requisitos, e, após resolver todas essas segmentações, uni-las para criar o app em sua totalidade.

-Reconhecimento de padrões (identificar similaridades e sequências em situações):

Identificamos padrões no contexto computacional a partir de similaridades e diferenças. O computador reconhece padrões por comparações, ao contrário do cérebro humano, que usa muitas vezes da subjetividade. O computador necessita de uma representação de atributos para que haja o aprendizado e o armazenamento de dados. E então, ocorrem as regras de decisão. O reconhecimento de padrões nada mais é do que a extração de características para que haja uma classificação de dados. Aplicações que utilizam reconhecimento de padrões: Machine Learning, Redes Neurais, Inteligência Artificial, Ciência de Dados, etc.

-Abstração (extrapolar o conceito do problema para uma forma generalista):

Processo intelectual de isolamento de um objeto da realidade. Durante esse processo também ocorre a generalização. Na computação, é muito utilizada em merge sort, clustering, busca binária, entre outros. Devemos analisar todas as características de um objeto e abstrair apenas aquelas relevantes à nossa resolução de problema atual, para que ocorra a generalização. Abstrações podem ser realizadas em formatos como listas, grafos e árvores. Até mesmo as próprias linguagens de programação em si podem ser vistas como um tipo de abstração, vez que passam algo do mundo real para conceitos abstratos gerais.

-Design de algoritmos (definir passo a passo da solução do problema):

O primeiro passo do design de algoritmos é o estudo e definição de dados de entrada e saída. O algoritmo descreve problemas por meio de ferramentas narrativas, fluxogramas ou pseudocódigos. Após a inserção dos dados, o algoritmo é codificado de acordo com a linguagem de programação escolhida. Para a criação de um algoritmo, devemos compreender o problema, definir os dados de entrada, definir o processamento, definir os dados de saída, utilizar um método de construção e realizar o teste e o diagnóstico.

Processo contínuo: Definir uma solução, testá-la e aperfeiçoá-la.

Competências extras:

-Pensamento sistemático;

-Melhora na colaboração dentro da equipe;

-Melhora na criatividade e design;

-Torna-se um facilitador.

Habilidades complementares: Raciocínio lógico e seu aperfeiçoamento.

O raciocínio lógico e seu aperfeiçoamento têm embasamento em indução (situações que nos fazem criar leis e teorias a seu respeito), dedução (fenômeno oposto à indução, que nos leva a prever situações através de leis e teorias) e abdução (conclusão que leva à uma premissa).

A partir de uma inferência, podemos seguir duas rotas: A sintética e a analítica. A primeira, nos leva a abdução e indução, enquanto a segunda nos leva à dedução.

O aperfeiçoamento do raciocínio lógico dentro da programação baseia-se no melhor uso de recursos (encontrar soluções eficientes e otimizar processos já existentes) e na melhora de códigos e algoritmos (simplificar linhas de códigos e estabelecer funções bem definidas).