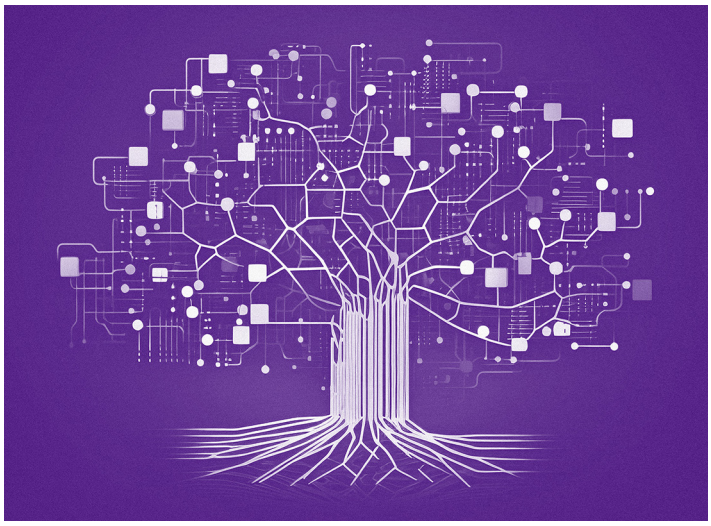
ESTRUTURA DE DADOS: ÁRVORES

MÓDULO 2

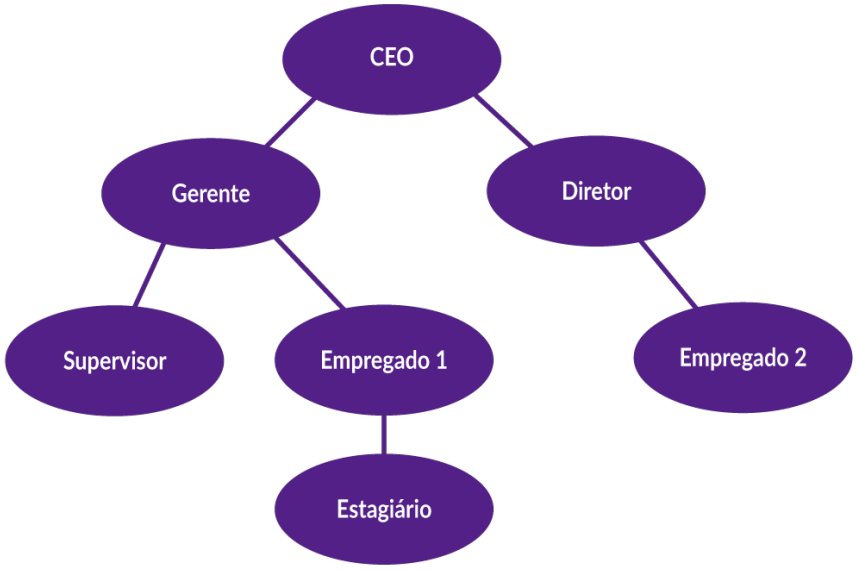
Existem vários tipos de árvores com diferentes características e aplicações, como árvores binárias, árvores de busca binária, árvores balanceadas, árvores AVL, árvores B, entre outras.



Com seus nós, a árvore de dados se torna uma estrutura complexa, mas hierárquica e organizada.

As **árvores** são amplamente **utilizadas em algoritmos de busca**, organização de dados, estruturas de banco de dados, processamento de linguagem natural, entre muitas outras áreas da ciência da computação.

Suponha que você esteja construindo um sistema de gerenciamento de uma empresa. Você pode usar uma árvore para representar a estrutura hierárquica da organização, onde cada nó representa um funcionário e suas relações com outros funcionários. Pense na seguinte estrutura:



A imagem mostra uma estrutura de dados em forma de árvore, com nós representados por ovais azuis. No topo da árvore está o nó “CEO”, que se ramifica em dois nós filhos: “Gerente” à esquerda e “Diretor” à direita. O nó “Gerente” se ramifica em dois outros nós: “Supervisor” e “Empregado 1”. Por sua vez, “Empregado 1” se ramifica em um nó filho chamado “Estagiário”. O nó “Diretor” tem um único nó filho chamado “Empregado 2”. Esta estrutura demonstra uma hierarquia típica de uma árvore, onde cada nó pode ter zero ou mais nós filhos, formando uma relação de parentesco entre os diferentes níveis.

Neste exemplo:

1. O CEO é o nó raiz da árvore, que não tem um nó pai.
2. O CEO tem dois filhos, o Gerente e o Diretor.
3. O Gerente tem três filhos: Supervisor, Empregado 1 e Empregado 2.
4. O Empregado 1 tem um filho, que é um Estagiário.

A utilização desta árvore permite uma representação eficiente da estrutura organizacional da empresa. Você pode facilmente realizar operações como encontrar o chefe de um funcionário específico, identificar todos os subordinados de um gerente, ou até mesmo navegar pela hierarquia para obter uma visão geral da organização.

Agora você pode testar isso no seu código, siga o passo a passo a seguir:

A. Para isso, será necessário instalar uma biblioteca no seu VSCode, chamada anytree.

No terminal do VSCode, digite: pip install anytree

```
PS C:\Users\Public\Senac_Codes> pip install anytree
```

B. E agora, volte ao código, e importe as classes Node e RenderTree.

- from anytree import Node, RenderTree

```
C: > Users > Public > Senac_Codes > Arvores_Senac.py > ...
  1    from anytree import Node, RenderTree
```

C. Agora você está preparado para começar sua estrutura. Insira os itens abaixo:

```
ceo = Node(“CEO”)

    gerente = Node(“Gerente”, parent=ceo)

    diretor = Node(“Diretor”, parent=ceo)

    supervisor = Node(“Supervisor”, parent=gerente)

    empregado1 = Node(“Empregado 1”, parent=gerente)

    empregado2 = Node(“Empregado 2”, parent=gerente)

    estagiario = Node(“Estagiário”, parent=empregado1)
```

```
C: > Users > Public > Senac_Codes > Arvores_Senac.py > ...
1   from anytree import Node, RenderTree
2
3   ceo = Node("CEO")
4   gerente = Node("Gerente", parent=ceo)
5   diretor = Node("Diretor", parent=ceo)
6   supervisor = Node("Supervisor", parent=gerente)
7   empregado1 = Node("Empregado", parent=gerente)
8   empregado2 = Node("Empregado", parent=gerente)
9   estagiario = Node("Estagiário", parent=empregado1)
```

Olhe, você está seguindo a mesma estrutura que criamos no organograma, agora falta pouco!

D. Para imprimir essa estrutura, digite as informações abaixo:

for pre, fill, node in RenderTree(ceo):

print("%s%s" % (pre, node.name))

```
C: > Users > Public > Senac_Codes > Arvores_Senac.py > ...
1   from anytree import Node, RenderTree
2
3   ceo = Node("CEO")
4   gerente = Node("Gerente", parent=ceo)
5   diretor = Node("Diretor", parent=ceo)
6   supervisor = Node("Supervisor", parent=gerente)
7   empregado1 = Node("Empregado 1", parent=gerente)
8   empregado2 = Node("Empregado 2", parent=gerente)
9   estagiario = Node("Estagiário", parent=empregado1)
10
11  for pre, fill, node in RenderTree(ceo):
12      print("%s%s" % (pre, node.name))
13
```

E. O resultado, após a execução, será:

```
PS C:\Users\Public\Senac_Codes>  c:;
' 'c:\Users\[redacted]\.vscode\ex
 'C:\Users\Public\Senac_Codes\Arvores
CEO
├── Gerente
│   ├── Supervisor
│   ├── Empregado 1
│   │   └── Estagiário
│   └── Empregado 2
└── Diretor
```

Perfeito, caro aluno!

Agora você conseguiu criar a estrutura em árvore, e imprimi-la. Se você quiser você pode inserir mais nós (para um novo funcionário neste caso), siga os passos a seguir:

A. Aqui um ponto importante: para adicionar o novo funcionário, você precisa usar a função **node**. A função possui dois parâmetros. Uma é o nome, e a outra é quem vai ser o pai desse nó, no caso é o diretor. Preste atenção nisso, caro aluno! É crucial para sua estrutura trabalhar corretamente.

- novo_funcionario = Node("Novo Funcionário", parent=diretor)

```
C: > Users > Public > Senac_Codes > Arvores_Senac.py > ...
1   from anytree import Node, RenderTree
2
3   ceo = Node("CEO")
4   gerente = Node("Gerente", parent=ceo)
5   diretor = Node("Diretor", parent=ceo)
6   supervisor = Node("Supervisor", parent=gerente)
7   empregado1 = Node("Empregado 1", parent=gerente)
8   empregado2 = Node("Empregado 2", parent=gerente)
9   estagiario = Node("Estagiário", parent=empregado1)
10
11  novo_funcionario = Node("Novo Funcionário", parent=diretor)
12
13  for pre, fill, node in RenderTree(ceo):
14      print("%s%s" % (pre, node.name))
```

B. Assim executado, temos o seguinte:

```
PS C:\Users\Public\Senac_Codes>
' 'c:\Users\[redacted]\.vsc
 'C:\Users\Public\Senac_Codes\A
CEO
├── Gerente
│   ├── Supervisor
│   ├── Empregado 1
│   │   └── Estagiário
│   └── Empregado 2
└── Diretor
    └── Novo Funcionário
```

C. E, para finalizar, se você quiser remover um nó, nós definiremos o atributo **parent** do nó para **None**:

- empregado1.parent = None

```
empregado1.parent = None
```

D. Executando o código:

C: > Users > Public > Senac_Codes > Arvores_Senac.py > ...

```python
from anytree import Node, RenderTree

ceo = Node("CEO")
gerente = Node("Gerente", parent=ceo)
diretor = Node("Diretor", parent=ceo)
supervisor = Node("Supervisor", parent=gerente)
empregado1 = Node("Empregado 1", parent=gerente)
empregado2 = Node("Empregado 2", parent=gerente)
estagiario = Node("Estagiário", parent=empregado1)

novo_funcionario = Node("Novo Funcionário", parent=diretor)

empregado1.parent = None

for pre, fill, node in RenderTree(ceo):
    print("%s%s" % (pre, node.name))
```

E. Você terá o resultado:

```
PS C:\Users\Public\Senac_Codes>
' 'c:\Users\[redacted]\.vsc
 'C:\Users\Public\Senac_Codes\A
CEO
├── Gerente
│   ├── Supervisor
│   └── Empregado 2
└── Diretor
    └── Novo Funcionário
```

O empregado 1 e o estagiário, que era seu filho, foram removidos.