# 04 - Trabalhando com Opcionais / Null Safety

## Variáveis Nullables

Dart reforça a segurança de nulos sólidos. Isso significa que valores não podem ser nulos a menos que você diga que eles podem ser. Em outras palavras, os tipos são padronizados como não anuláveis.

Por exemplo, com segurança nula, esse código retorna um erro. Uma variável do tipo int não pode ter o valor nulo:

```
int a = null; // INVALIDO.
```

Ao criar uma variável, adicione ? ao tipo para indicar que a variável pode ser nula:

```
int? a = null; // VALIDO.
```

Você pode simplificar um pouco esse código porque, em todas as versões do Dart, nulo é o valor padrão para variáveis não inicializadas

```
int? a; // O valor inicial de a é nulo.
```

Nullables existem para prevenir erros de referência nula em tempo de execução, conhecidos como Null Pointer Exceptions. Ao declarar explicitamente quais variáveis podem ser nulas, o compilador ajuda a identificar possíveis falhas, aumentando a segurança e a robustez do código.

Acessando uma variável nullable com verificação:

```
void main() {
  String? nome;
  if (nome != null) {
    print(nome.length);
  } else {
    print('Nome não foi fornecido.');
  }
}
```

Usando o operador de acesso condicional ?:

```
void main() {
  String? nome = 'Alice';
  print(nome?.length); // Saída: 5
}
```

Utilizando o operador de coalescência nula ??:

```
void main() {
  String? nome;
  print(nome ?? 'Usuário anônimo'); // Saída: Usuário anônimo
}
```

Temos que evitar o uso do operador de assertiva não nula (!) sem necessidade, pois ele força o compilador a tratar uma variável nullable como não nula, podendo causar erros em tempo de execução se a variável for null.

```
void main() {
  String? nome;
  print(nome!.length); // Erro em tempo de execução se 'nome' for null
}
```

Levando em consideração as boas práticas, devemos sempre seguir:

- Nullables permitem que variáveis aceitem o valor null, aumentando a expressividade e segurança do código.
- Eles existem para prevenir erros de referência nula, comuns em linguagens que não possuem null safety.
- Para utilizar, declare o tipo com ? e use operadores como ?, ?? para manipulação segura.
- Evite o uso excessivo de !, pois pode mascarar problemas e introduzir erros em tempo de execução.