

# Quer deixar seu código ABAP mais limpo? Descubra como usar **Template Strings** agora!

---

Hoje trago uma dica útil e moderna para quem trabalha com **ABAP**: o uso de **Template Strings** para deixar o código mais limpo, legível e poderoso.

## O que são Template Strings no ABAP?

As **Template Strings** são uma funcionalidade moderna do **ABAP** que permite construir textos dinâmicos de forma muito mais simples, legível e eficiente. Em vez de usar várias concatenações manuais ou comandos como **CONCATENATE**, você escreve o texto diretamente com variáveis embutidas no meio da string.

💡 Principais vantagens de usar Template Strings:

- Código mais limpo: reduz complexidade visual e linhas de concatenação.
- Facilidade de manutenção: alterar o formato do texto fica muito mais rápido e intuitivo.
- Formatação automática: com parâmetros como **ALPHA**, **CURRENCY**, **DATE** e **TIME**, você já insere os valores formatados diretamente.
- Expressões integradas: você pode fazer pequenos cálculos ou manipulações dentro da própria string.
- Redução de erros: menos comandos separados significa menos risco de esquecer espaços, separadores ou formatações.

✅ **Exemplo básico:** Exemplo básico para o uso do template string.

```
* Exemplo Básico
DATA(lv_nome)   = 'Yuri'.
DATA(lv_cidade) = 'Rio de Janeiro'.
DATA(lv_msg)    = |Olá, meu nome é { lv_nome } e moro no { lv_cidade }|.

out->write( lv_msg ).
ENDMETHOD.
```

**Resultado esperado:**

```
Olá, meu nome é Yuri e moro no Rio de Janeiro.
```

---

🔧 **Funções dentro da Template String:** Também é possível utilizar funções dentro da template string, vamos ver como fica?!

```
* Uso de funções
DATA(lv_spaces) = '  Variável  '.
lv_msg = |Esta "{ lv_spaces }" está com espaços no início e no fim, mas podemos consertar usando a função condense, olha como ficou legal! "{ condense( lv_spaces ) }"|.
out->write( lv_msg ).
```

**Resultado esperado:**

Esta " Variável " está com espaços no início e no fim, mas podemos consertar usando a função condense, olha como ficou legal! "Variável".

🔧 **Formatação ALPHA e CURRENCY:** É possível usar algumas conversões diretamente na string, como é o caso das conversões Alpha e Currency.

```
* A moeda do Brasil usa 2 decimais em sua representação
* Já a moeda da Hungria(HUF) não usa decimais na representação de sua moeda
DATA(lv_resultado) = |Código: { condense( | { lv_codigo ALPHA = OUT } | ) }, Preço: { lv_preco CURRENCY = 'BRL' }, Preço(Hungria): { lv_preco CURRENCY = 'HUF' }|.
out->write( lv_resultado ).
```

**Resultado esperado:**

Código: 1234, Preço: 1234.56. Preço(Hungria): 123456

📅 **Formatação DATE = USER e ENVIRONMENT:** Esse realmente ajuda, é o fim do código cheio de "IFs" para imprimir a formatação correta para o usuário!

```
* Formatação DATE = USER e ENVIRONMENT
* Meu usuário é americano, portanto ao usar DATE = USER, a data ficará MM/DD/YYYY
* Já o ambiente de testes é brasileiro, portanto ao usar DATE = ENVIRONMENT, a data ficará DD/MM/YYYY
DATA(lv_data_atual) = |Data atual: { cl_abap_context_info=>get_system_date( ) DATE = USER }. Data atual(Brasil): { cl_abap_context_info=>get_system_date( ) DATE = ENVIRONMENT }|.
out->write( lv_data_atual ).
```

**Resultado esperado:**

Data atual: 04/26/2025. Data atual(Brasil): 04/26/2025

🕒 **Formatação TIME = ISO:** Esse daqui vai te ajudar a formatar a hora de maneira bem rápida..

```
* Formatação TIME = ISO
data(lv_hora_atual) = |Hora atual: { cl_abap_context_info=>get_system_time( ) TIME = ISO }. Hora sem conversão: { cl_abap_context_info=>get_system_time( ) }|.
out->write( lv_hora_atual ).
```

**Resultado esperado:**

Hora atual: 16:52:01. Hora sem conversão: 165201

🧠 **Expressões dentro da string:** Outra funcionalidade interessante, é a capacidade de usar expressões diretamente dentro do string

```
* Expressões dentro da string
DATA(lv_expressao) = |A soma de 2 + 7 = { 2 + 7 }. Já o produto 2 x 4 = { 2 * 4 }|.
out->write( lv_expressao ).
```

**Resultado esperado:**

A soma de  $2 + 7 = 9$ . Já o produto  $2 \times 4 = 8$ .

---

### Exemplo de Multilinhas:

```
Exemplo de Multilinhas
lv_msg = |Olá, vocês sabiam dessa? | &
        |Sejam bem vindos ao mundo moderno do ABAP!|.
out->write( lv_msg ).
```

### Resultado esperado:

Olá, vocês sabiam dessa? Sejam bem vindos ao mundo moderno do ABAP!

---


### Escape de chaves literais:


```
* Escape de chaves literais com \
lv_msg = |Se vocês quiserem colocar uma chave { literal, será necessário o uso do caractere de scape \\. \nEsse é um extra, percebu que eu apareci na linha de baixo}|.
out->write( lv_msg ).
```

### Resultado esperado:

Se vocês quiserem colocar uma chave { literal, será necessário o uso do caractere de scape \.  
Esse é um extra, percebu que eu apareci na linha de baixo?

---

 Se curtiu esse conteúdo, comenta ou compartilha!

 E me segue pra mais dicas práticas sobre ABAP!