

# Dominando Tipos de Dados Globais no ABAP

A base para aplicações Fiori robustas, traduzíveis e de fácil manutenção.

# O Dilema do Desenvolvedor: "Por que não usar apenas `abap.char(20)`?"

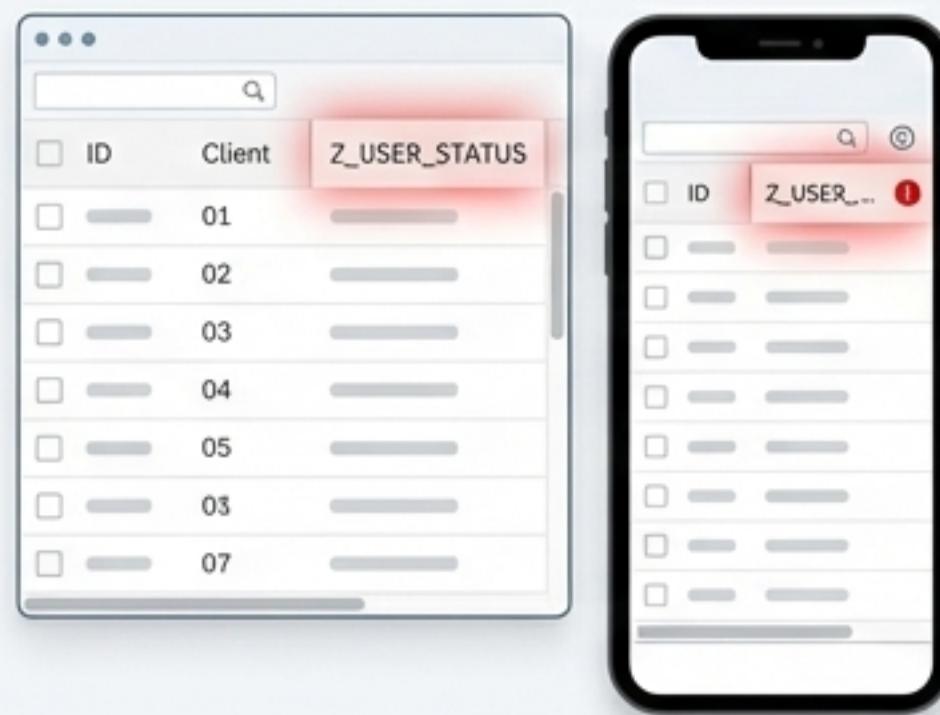
No desenvolvimento ágil, a velocidade parece essencial. Definir um tipo diretamente na tabela é o caminho mais rápido para ter um campo funcional. Mas qual é o custo real desse atalho a longo prazo?

```
DEFINE TABLE zmy_table {  
    key client      : abap.clnt;  
    key id          : abap.numc(10);  
    z_user_status : abap.char(20);  
}
```

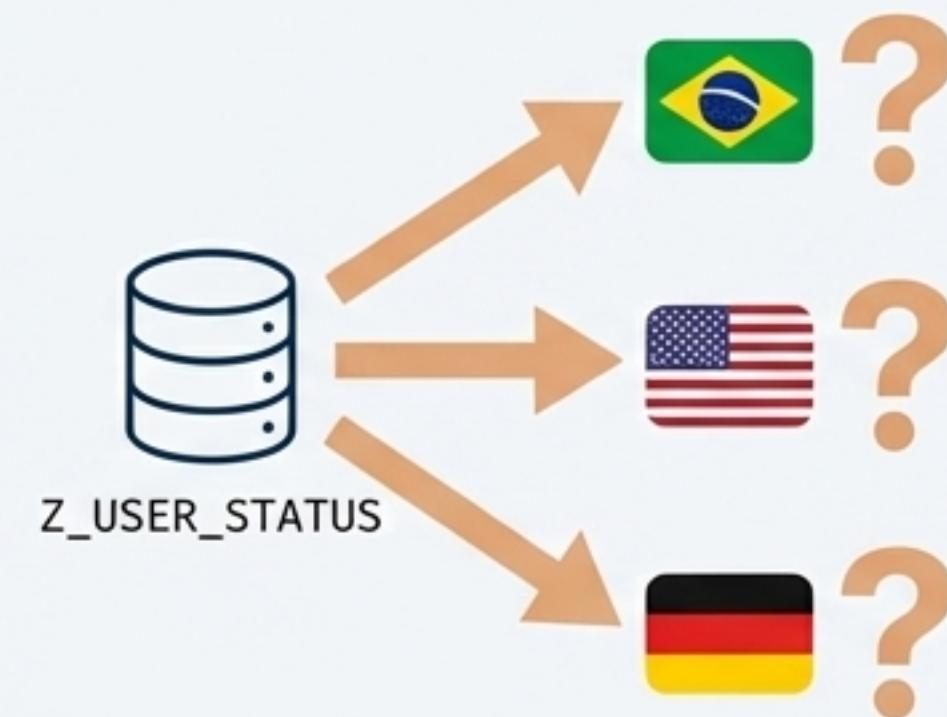


# Quando o Atalho Cobra seu Preço: O Pesadelo da Manutenção

## UI Quebrada

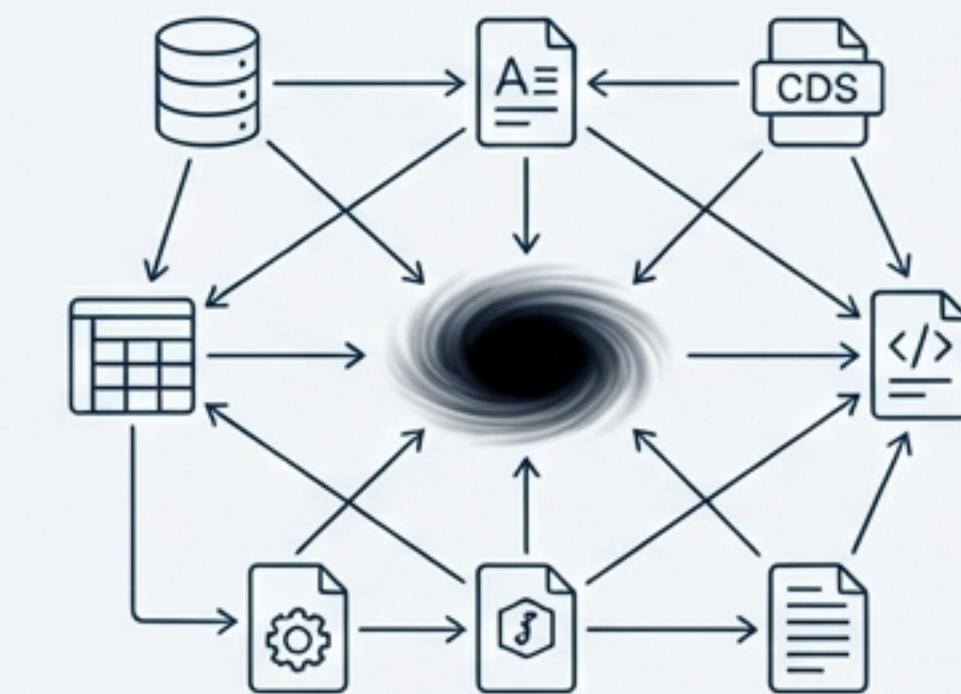


## Caos na Tradução



O mesmo rótulo precisa ser traduzido manualmente em 10 lugares diferentes.

## Análise de Impacto Cega



Onde mais este conceito de negócio é usado?  
Impossível saber com um tipo primitivo.

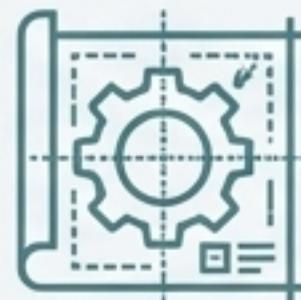
**Um campo, múltiplos problemas. Multiplique por 100 campos.**

# A Solução Estratégica: O Princípio da Fonte Única da Verdade

O ABAP Dictionary desacopla a *\*definição* do *\*uso\**. Isso garante que a semântica, os textos e as traduções de um campo sejam definidos UMA VEZ e reutilizados consistentemente em TODOS os lugares.



# A Arquitetura de Duas Camadas: A Base da Excelência



## O TÉCNICO: DOMÍNIO

Define as propriedades puramente técnicas.



- Tipo de Dado e Tamanho  
(CHAR, DEC, etc.)



- Propriedades de Saída  
(ex: Permite minúsculas)



- Rotinas de Conversão

*Como o dado é armazenado e formatado?*



## O SEMÂNTICO: ELEMENTO DE DADOS

Adiciona significado de negócio.



- Rótulos de Campo  
(Curto, Médio, Longo)



- Ajuda de Pesquisa (Value Help)



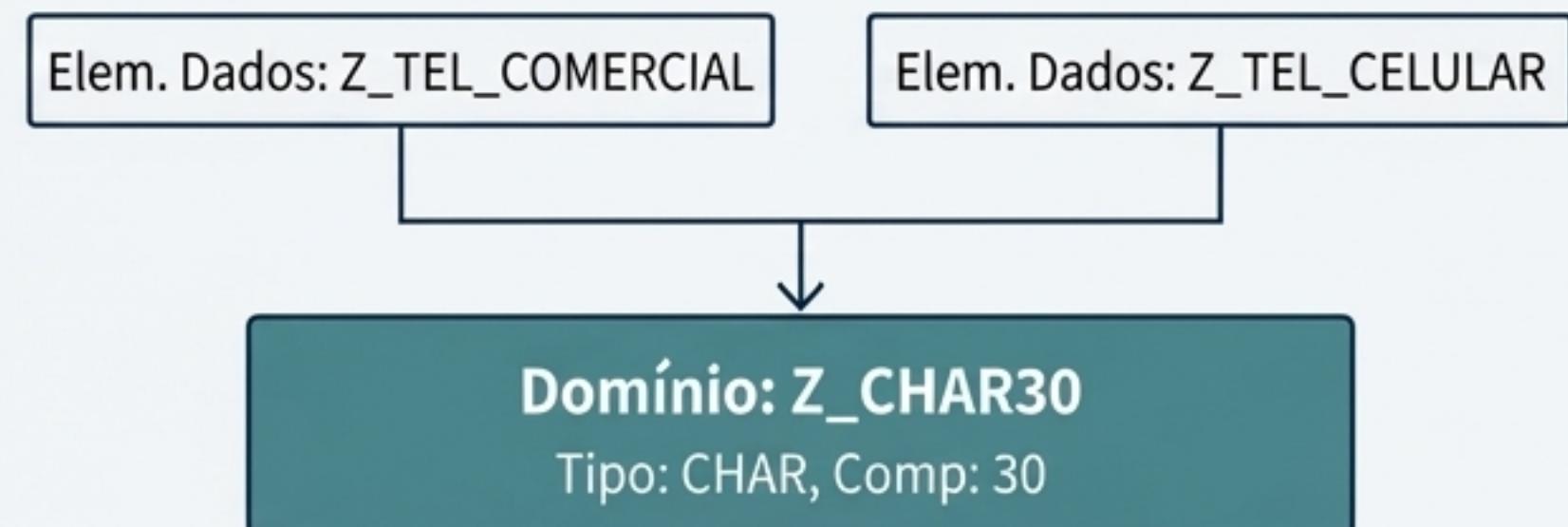
- Documentação (Ajuda F1)

*O que este dado significa para o usuário?*

# Em Foco: O Domínio, a Fundação Técnica Reutilizável

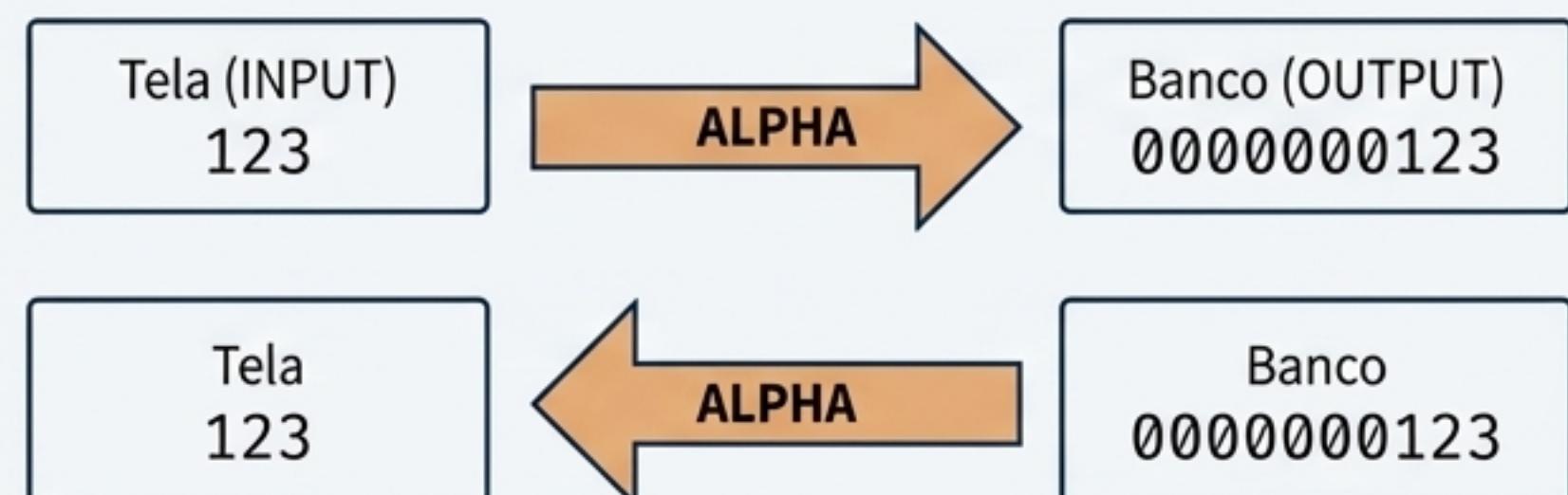
## Reutilização

Vários Elementos de Dados ('Telefone Comercial', 'Telefone Celular') podem compartilhar o mesmo Domínio técnico (ex: `Z\_CHAR30`).



## O Poder da Rotina de Conversão

A rotina `ALPHA` garante a consistência de chaves numéricas sem esforço do desenvolvedor.



Define uma função que é executada automaticamente ao mover dados da tela para o banco (INPUT) e vice-versa (OUTPUT).

# O Elemento de Dados: O Segredo da UI Perfeita no Fiori

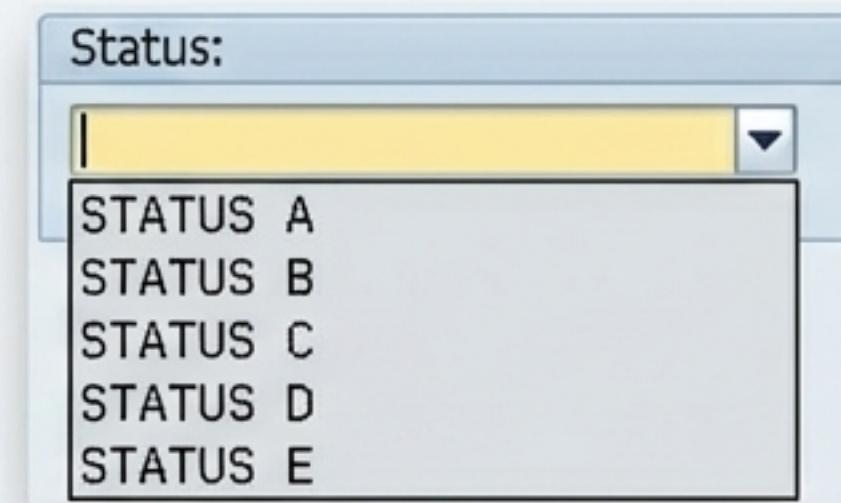
Você define os rótulos (Curto, Médio, Longo, Cabeçalho) uma única vez. O Fiori Elements escolhe o melhor para cada dispositivo, **automaticamente**. Isso é **design responsivo inteligente**.



*Se você não preencher isso corretamente,  
a UI ficará quebrada em telas pequenas.*

# A Evolução dos Valores Fixos: De Listas Estáticas para Código Tipado

## O Clássico: Valores Fixos de Domínio



- + Validação automática em telas antigas (Dynpro).
- Não há segurança de tipo (type-safety) no código ABAP. O compilador não ajuda a prevenir erros lógicos com `magic strings`.

## O Moderno: Enumerações (Enums)



- + Segurança de tipo (à prova de erros), código mais legível e auto-documentado.



# Apresentando as Enumerações: Código Legível e à Prova de Erros

Defina um conjunto **finito de valores nomeados**. Uma **variável tipiada** com um Enum só pode aceitar os valores definidos nele. O código se torna auto-documentado e o compilador se torna seu aliado.

```
INTERFACE if_statusEnums PUBLIC.
```

TYPES:

```
BEGIN OF ENUM ty_Status STRUCTURE status BASE TYPE char1,  
    created    VALUE 'C',  
    processed  VALUE 'P',  
    rejected   VALUE 'R',
```

```
END OF ENUM ty_Status STRUCTURE status.
```

```
ENDINTERFACE.
```

 Mapeia o nome semântico (legível no código) para o valor técnico (armazenado).

 Define o tipo técnico subjacente que será armazenado no banco de dados.

# O Compilador Como Seu Guardião

```
DATA: lv_status TYPE if_statusEnums=>ty_status.  
  
lv_status = if_statusEnums=>status-rejected.  
  
IF lv_status = if_statusEnums=>status-rejected.  
...  
ENDIF.
```



```
DATA: lv_status TYPE if_statusEnums=>ty_status.  
  
lv_status = 'X'.
```

**ERRO DE SINTAXE!**

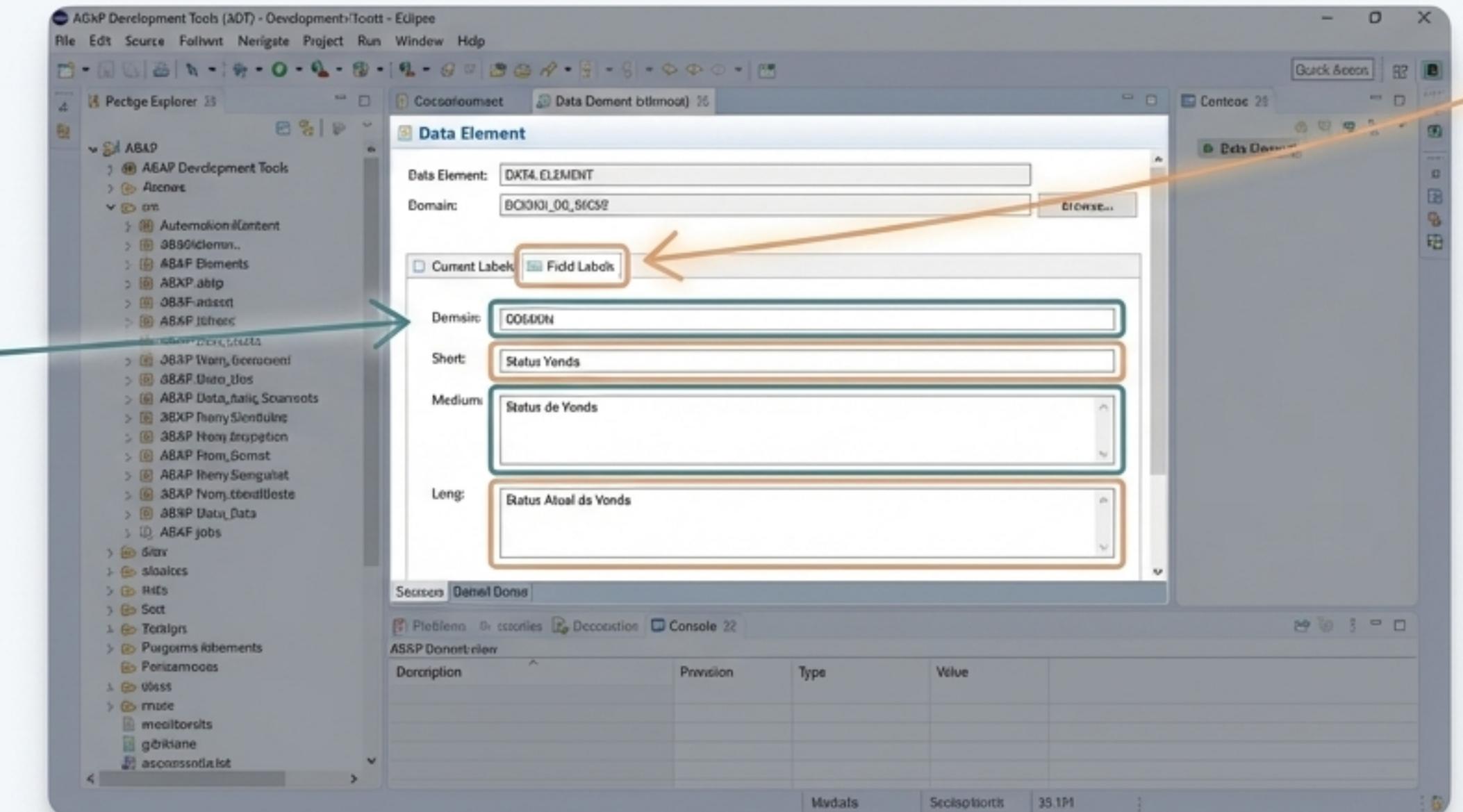


Código claro, seguro e sem ‘magic strings’.

Atribuições inválidas são impossíveis. O erro é pego em tempo de compilação, não em produção.

# Da Teoria à Prática no Eclipse (ADT)

Aqui você conecta  
a base TÉCNICA.



**Dica:** Embora seja um editor de formulário, ele está totalmente integrado ao seu projeto e ao controle de versão (Git).

# A Estrutura Completa: Uma Visão Geral



Tabela: `ZCLIENTE-TEL\_COMERCIAL`



Elemento de Dados: `Z\_TELEFONE\_COMERCIAL`

Semântica: 'Telefone Comercial', 'Tel. Comercial do Cliente'

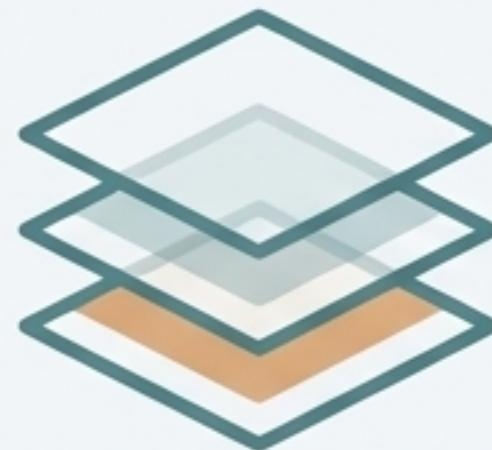


Domínio: `Z\_CHAR30`

Técnica: CHAR(30), case-insensitive

Cada camada tem sua responsabilidade, criando um sistema desacoplado, reutilizável e fácil de manter.

# Diretrizes Para o Sucesso



## PENSE EM CAMADAS:

Separe sempre a definição técnica (Domínio) da semântica (Elemento de Dados).



## ADOTE ENUMS:

Para valores fixos, use Enumerações. São a abordagem moderna, segura e legível.



## PRIORIZE OS RÓTULOS:

O preenchimento correto dos Field Labels é o que diferencia uma UI amadora de uma profissional e responsiva.



## INVISTA TEMPO AGORA:

O tempo gasto na modelagem de dados correta é pago com juros na fase de manutenção e evolução do seu aplicativo.