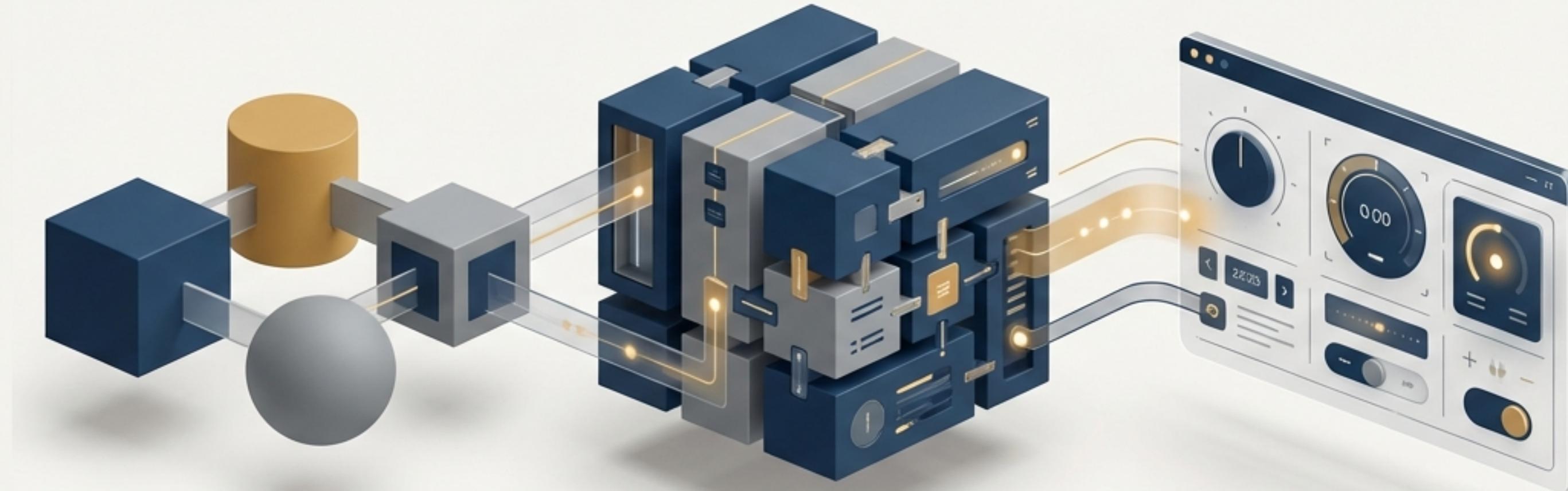


Dominando o SAP CAP CDS

Dos Fundamentos ao Controle Fino



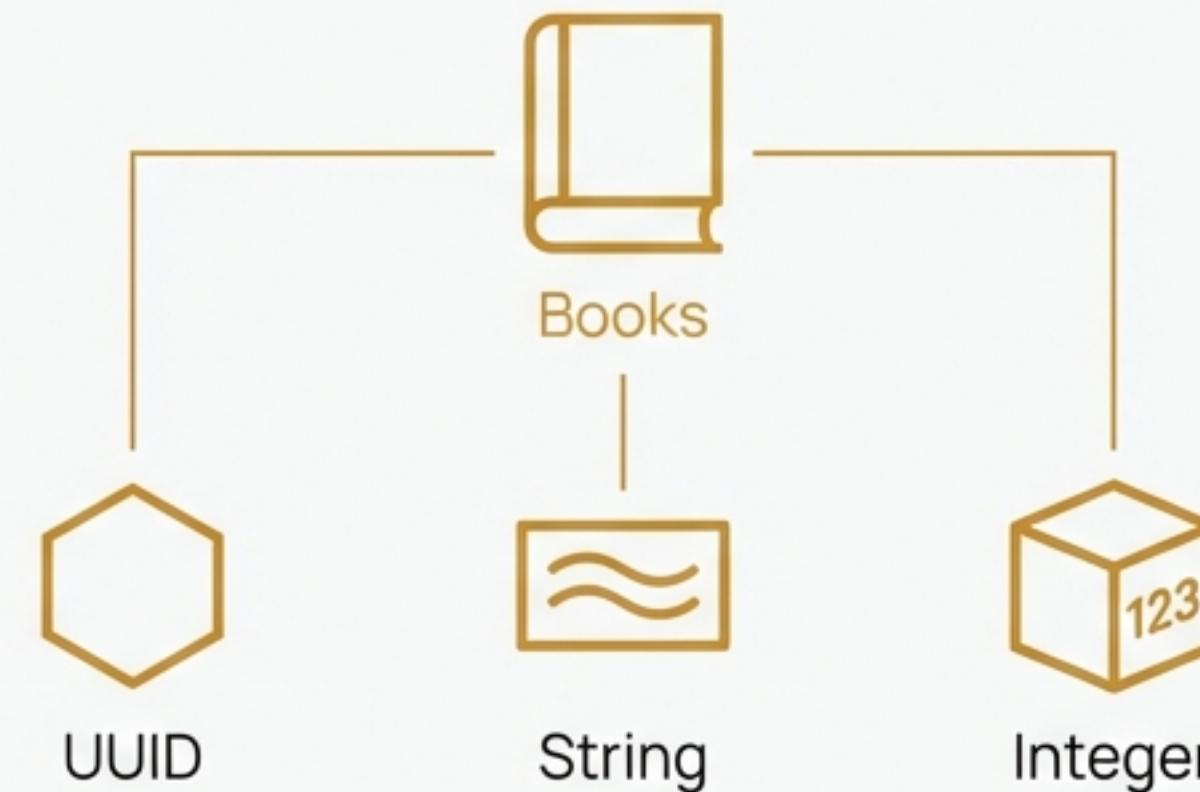
Um guia visual para desenvolvedores sobre como construir aplicações robustas e eficientes com os principais componentes do Core Data Services.

Os Fundamentos: Tipos de Dados Essenciais do CDS

Todo modelo de dados robusto começa com a definição precisa de seus elementos. O CDS oferece um conjunto rico de tipos de dados integrados que são mapeados para os tipos SQL correspondentes no banco de dados, garantindo uma base sólida e previsível para suas entidades.

```
// Um exemplo simples para contextualizar os tipos
entity Books {
    key ID : UUID;
    title: String(111);
    stock: Integer;
    price: Price; // Exemplo de tipo customizado
}

type Price : Decimal;
```



O Arsenal de Tipos: Numéricos e Temporais

Tipos Numéricos

Integer (Int32)

Mapeado para `INTEGER`

Int16

Inteiro de 16 bits, mapeado para `SMALLINT`

Int64

Inteiro de 64 bits, mapeado para `BIGINT`

UInt8

Inteiro de 8 bits sem sinal, mapeado para `TINYINT`

Decimal(prec, scale)

Mapeado para `DECIMAL`

Double

Ponto flutuante, mapeado para `DOUBLE`

Tipos Temporais

Date

Ex: `2022-12-31`. Mapeado para `DATE`.

Time

Ex: `23:59:59`. Mapeado para `TIME`.

DateTime

Precisão de segundos, mapeado para `TIMESTAMP`.

Timestamp

Precisão de microssegundos, mapeado para `TIMESTAMP`.

(1) O mapeamento concreto pode variar dependendo do banco de dados específico.

O Arsenal de Tipos: Texto, Binários e Especiais

Tipo CDS	Observações	SQL ANSI
UUID	CAP gera UUIDs compatíveis com RFC 4122.	`NVARCHAR(36)`
Boolean	Valores: `true`, `false`, `null`.	`BOOLEAN`
String(length)	Comprimento padrão no HANA: 5000. (4)	`NVARCHAR`
Binary(length)	Comprimento padrão no HANA: 5000. (4)	`VARBINARY`
LargeString	Dados ilimitados, geralmente via streaming.	`NCLOB`
LargeBinary	Dados ilimitados, geralmente via streaming.	`BLOB`
Map	Mapeado para `NCLOB` no HANA. Tipo JSON.	`NCLOB`



Dica de Produção

(4) "Aplicações produtivas devem sempre usar um comprimento explícito. Use o padrão apenas para prototipagem rápida."

O Acelerador: Por Que Usar `@sap/cds/common`?

Em vez de reinventar a roda, o SAP CAP oferece uma **biblioteca pré-construída**, `@sap/cds/common`, que encapsula **padrões e boas práticas**. Usá-la não é apenas uma conveniência, é uma estratégia para construir aplicações melhores e mais rápido.



Modelos Concisos e Compreensíveis: Reduza o código repetitivo e foque na lógica de negócio.



Promove a Interoperabilidade: Use tipos e estruturas comuns que funcionam em todo o ecossistema.



Boas Práticas Comprovadas: Incorpore design patterns testados em aplicações reais.



Suporte Automático a Dados Localizados: Simplifique a complexidade da internacionalização.



Extensibilidade Integrada: A biblioteca é um ponto de partida, não uma limitação.

O resultado é direto, capturando as melhores práticas que aprendemos com aplicações de negócio reais, com pegada mínima, desempenho otimizado e máxima adaptabilidade e extensibilidade.

O Poder dos Aspects: Simplificando o Código com `cuid` e `managed`

Como os aspects fornecem atalhos para padrões de modelagem canônicos.

aspect `cuid` - Chaves Primárias Universais

O Jeito Verboso

```
entity Foo {  
    key ID : UUID;  
    // ...  
}
```



O Jeito `cuid`

```
entity Foo : cuid {  
    // ...  
}
```

O aspect `cuid` adiciona uma chave primária `UUID` canônica, que os runtimes preenchem automaticamente.

aspect `managed` - Campos de Auditoria Automáticos

O Jeito Verboso

```
entity Foo {  
    createdAt : Timestamp @cds.on.insert: $now;  
    createdBy : User @cds.on.insert: $user;  
    modifiedAt : Timestamp @cds.on.insert: $now @cds.on.update: $now;  
    modifiedBy : User @cds.on.insert: $user @cds.on.update: $user;  
    // ...  
}
```



O Jeito `managed`

```
entity Foo : managed {  
    // ...  
}
```

O aspect `managed` adiciona e gerencia automaticamente os quatro campos de criação e modificação.

O Padrão de Reúso: Tipos e Listas de Códigos

A biblioteca `common` simplifica o uso de entidades comuns como Países, Moedas e Idiomas através de um padrão elegante: um tipo simples que é uma associação a uma lista de códigos completa e extensível.

1

Definindo o Tipo de Reúso (`Country`)

```
type Country : Association to sap.common.Countries;
```

`Country` não é um tipo primitivo, mas um atalho para se associar à entidade `Countries`.

2

Usando o Tipo no seu Modelo (`Addresses`)

```
using { Country } from '@sap/cds/common';
entity Addresses {
    street : String;
    town   : String;
    country : Country; // Simples e legível
}
```

O tipo `Country` é usado na entidade `Addresses`, mantendo o código limpo e legível.

3

```
CREATE TABLE Addresses (
    street NVARCHAR(5000),
    town  NVARCHAR(5000),
    country_code NVARCHAR(3) -- Chave estrangeira
);
```

O CDS resolve a associação, criando uma coluna de chave estrangeira (`country_code`) que referencia a tabela `Countries`.

Por Dentro das Listas de Códigos

As entidades de lista de códigos são projetadas para serem minimalistas, mas baseadas em padrões internacionais, oferecendo uma base sólida para dados mestres.

`'aspect sap.common.CodeList'` (A Base de Tudo)

```
aspect sap.common.CodeList {  
    name : localized String(111);  
    descr : localized String(1111);  
}
```

`entity sap.common.Countries : CodeList`

```
cds  
    key code : String(3); // ISO 3166-1 alpha-2  
    ou alpha-3
```

`entity sap.common.Currencies : CodeList`

```
cds  
    key code : String(3); // ISO 4217 alpha-3  
    symbol : String(5);  
    minorUnit : Int16;
```

`entity sap.common.Languages : CodeList`

```
cds  
    key code : sap.common.Locale; // Ex: en_GB  
}
```

A Mágica do `localized`: Internacionalização Simplificada

Adicionar a palavra-chave `localized` a um elemento dispara um poderoso mecanismo no CDS que gera automaticamente as tabelas e views necessárias para gerenciar traduções de forma eficiente.

Sua Definição Simples

```
cds
entity Foo {
    descr: localized String;
}
```

O CDS Gera...

Geração da Tabela de Textos (`_texts`)

O CDS cria uma tabela `Foo_texts` para armazenar as traduções.

```
sql
CREATE TABLE Foo_texts (
    ID NVARCHAR(36),
    locale NVARCHAR(14),
    descr NVARCHAR(1000),
    PRIMARY KEY(ID, locale)
);
```

E Também Gera...

Geração da View Localizada

Uma view é criada para unir a tabela principal com as traduções, selecionando o idioma correto automaticamente.

```
sql
CREATE VIEW localized_Foo AS SELECT
    COALESCE(localized.descr, descr) AS descr
FROM Foo LEFT JOIN Foo_texts AS localized
ON Foo.ID = localized.ID
AND localized.locale = session_context('locale')
```

Com uma única palavra-chave, você obtém uma solução completa de persistência para dados multilíngues, com fallback automático multilíngues, com fallback automático.

Adapte e Estenda: Faça a Biblioteca Trabalhar para Você

Os modelos de `@sap/cds/common` são intencionalmente minimalistas. Use os recursos padrão do CDS, como `extend` e `aspects`, para adaptá-los às suas necessidades específicas.

Adicionando Detalhes a `Countries`

Adicionar mais campos do padrão ISO 3166-1.

```
using { sap.common.Countries } from '@sap/cds/common';

extend Countries {
    alpha3 : String(3);      // Código de 3 letras
    numcode : Integer;       // Código numérico
    status : String(111);
}
```

Criando uma Nova Lista de Códigos para `Regions`

Criar uma nova entidade para regiões (ISO 3166-2) e associá-la a `Countries`.

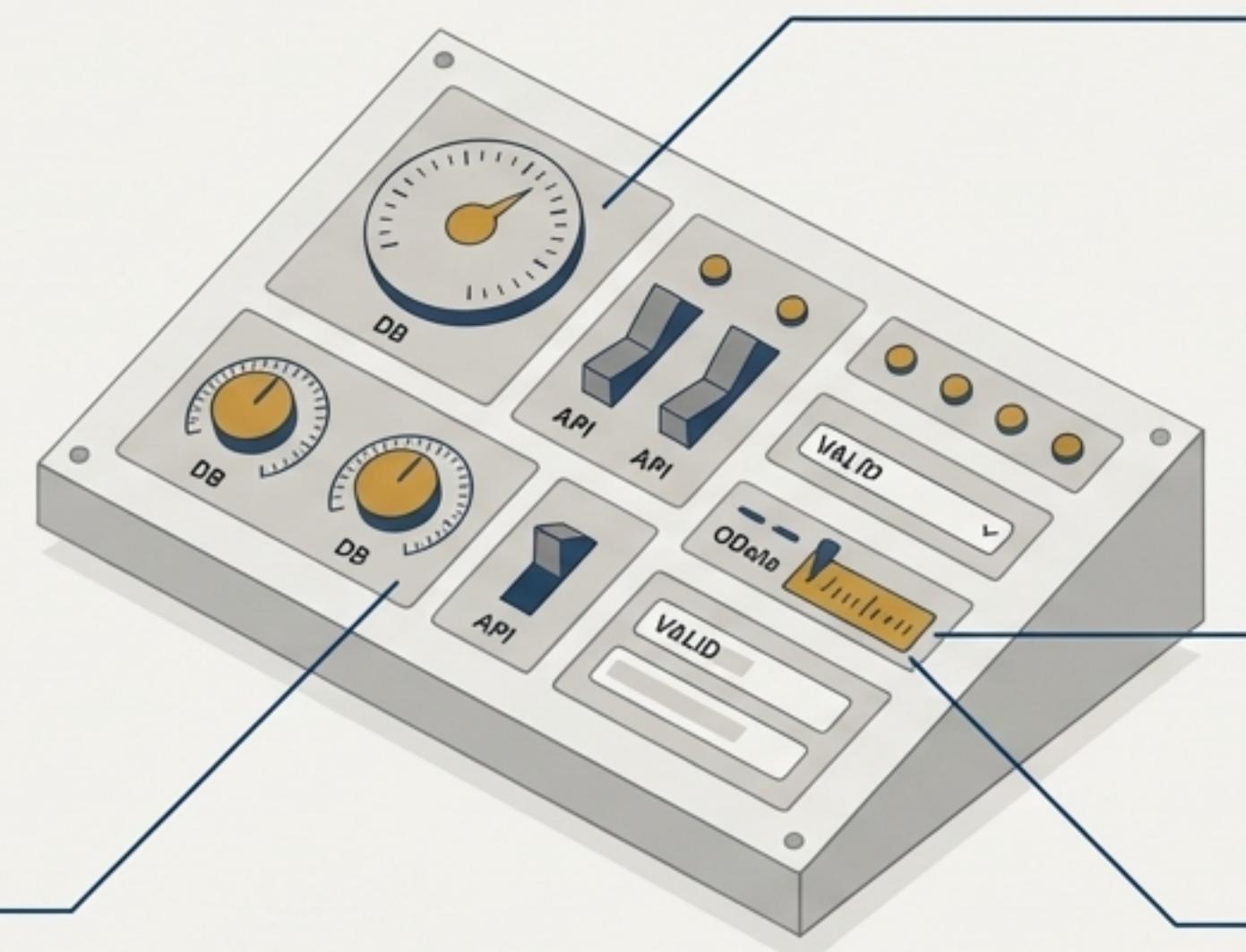
```
using sap from '@sap/cds/common';

// 1. Defina a nova entidade
entity Regions : sap.common.CodeList {
    key code : String(5); // Ex: DE-BW
    country : Association to sap.Countries;
}

// 2. Crie a associação bidirecional
extend sap.Countries {
    regions : Composition of many Regions on regions.country = $self;
}
```

O Painel de Controle: Ajuste Fino com Anotações

Uma vez que seu modelo está estruturado com tipos e aspects, as anotações permitem que você controle precisamente como ele se comporta. Elas são metadados que instruem o compilador e os runtimes do CDS a modificar o comportamento padrão na geração de DDL, na exposição de serviços OData e na validação de dados.



Persistência: Controle como as entidades são mapeadas para o banco de dados.

Serviços / APIs: Defina o comportamento dos seus endpoints de serviço.

Validação de Entrada: Aplique regras de validação diretamente no modelo.

OData: Influencie a geração de metadados e o comportamento dos serviços OData.

Guia de Anotações Essenciais: Persistência e Serviços

Persistência

`@cds.persistence.skip`

Impede que o CDS gere uma tabela/view no banco de dados para esta entidade. Ideal para views que são apenas para serviços.

```
entity MyView as ... @cds.persistence.skip;
```

`@cds.persistence.exists`

Informa ao CDS que a tabela/view correspondente já existe no banco de dados.

`@cds.on.insert / @cds.on.update`

Usado pelo aspect `managed` para definir valores em operações de criação/atualização.

```
createdAt : Timestamp @cds.on.insert: $now;
```

Serviços / APIs

`@cds.autoexpose`

Expõe automaticamente uma entidade em qualquer serviço OData que tenha uma associação a ela. Padrão nas listas de códigos `common`.

`@readonly`

Torna um elemento ou uma entidade inteira somente leitura no nível do serviço.

```
element : String @readonly;
```

`@cds.search`

Permite que o campo seja utilizado em buscas full-text (`\$search`) no serviço.

```
title : String @cds.search;
```

Guia de Anotações Essenciais: Validação e OData



Validação de Entrada

`@mandatory`

Garante que o campo deve ter um valor.

```
name : String @mandatory;
```

`@assert.format`

Valida o valor do campo contra uma expressão regular.

```
email: String @assert.format: '...';
```

`@assert.range`

Garante que um valor numérico esteja dentro de um intervalo.

```
rating: Integer @assert.range: [1, 5];
```



OData

`@odata.etag`

Habilita o controle de concorrência otimista (ETags) para uma entidade.

`@Core.Immutable` / `@Core.Computed`

Informa ao cliente UI que um campo não pode ser alterado ou é calculado no backend.

`@title` e `@description`

Fornecem rótulos e descrições legíveis para UIs, que são refletidos nos metadados do serviço.

```
bookTitle : String @title: 'Título do Livro';
```

`@ValueList.entity`

Anotação de UI para associar uma ajuda de valores (value help) a um campo.

Montando o Quebra-Cabeça: Um Modelo Completo em Ação

Vamos aplicar tudo o que aprendemos. Este exemplo de uma entidade `PurchaseOrders` combina tipos de dados, aspects da biblioteca `common`, tipos de reúso (padrão e customizado) e anotações para criar um modelo de dados rico e funcional.

```
using { Country, Currency } from '@sap/cds/common';
using { Region } from './your-common'; // Nosso tipo customizado

entity PurchaseOrders : cuid, managed { // ACT II: Aspects
    orderNumber : String(10) @readonly; // ACT III: Anotação de serviço
    orderDate   : Date @mandatory; // ACT I: Tipo + ACT III: Anotação
    supplier    : Association to Suppliers;
    totalAmount : Decimal(15, 2); // ACT I: Tipo de dado
    currency    : Currency; // ACT II: Tipo de reúso
    shipToAddress : {
        street : String;
        city   : String;
        region : Region; // ACT II: Tipo de reúso customizado
        country: Country; // ACT II: Tipo de reúso
    }
}
@cds.persistence.skip // ACT III: Anotação de persistência
entity ActiveOrders as projection on PurchaseOrders where ...;
```

Fundamentos

(Tipos): Os blocos de construção básicos do nosso modelo.

Acelerador

(@sap/cds/common): Usando aspects e tipos de reúso para código conciso e padronizado.

Controle (Anotações): Ajustando o comportamento do modelo para persistência, serviços e validação.