

Dominando Dados Temporais com SAP CAP

Uma Jornada do Modelo de Dados à Análise Histórica

Um guia prático para construir aplicações empresariais que gerenciam a validade de dados de forma nativa.



A Filosofia Central: Capture a Intenção, Não o Como

O SAP CAP foca no modelo conceitual. Descrevemos **o que** queremos, e o framework cuida de **como** implementar da melhor forma. Isso nos permite focar na lógica de negócios em vez de código repetitivo.

Capture Intent — What, not How!

```
using { cuid, managed }
from '@sap/cds/common';

entity Books : cuid, managed {
    title : localized String;
    author : Association to Authors;
}
```

Aspectos pré-definidos para funcionalidades comuns (chaves e auditoria).

Declaração de intenção para dados traduzidos.

O código declarativo permite ao CAP otimizar a implementação para o banco de dados e gerar serviços robustos automaticamente.

Passo 1: A Fundação — Modelagem de Domínio com CDS

Modelos limpos, planos e compreensíveis são a base de qualquer grande aplicação. O CDS torna essa tarefa simples e direta.



```
entity Books : cuid {  
    title : String;  
    author : Association to Authors;  
}  
  
entity Authors : cuid {  
    name : String;  
    books : Association to many Books on books.author = $self;  
}
```

Entidades e Chaves

- Como definir as estruturas centrais de dados.
- Melhor prática: Use UUIDs para chaves primárias (`key ID: UUID;`). Isso garante unicidade universal e facilita a distribuição.

Associações

- Como conectar entidades de forma inteligente (`Association to...`).
- Prefira associações gerenciadas (`managed associations`) para manter os modelos concisos.

Convenções de Nomenclatura

- **Entidades:** Nomes capitalizados e no plural (ex: `Books`, `Authors`).
- **Elementos:** Nomes em minúsculas (ex: `title`, `name`).
- **Concisão:** Prefira `Authors.name` em vez de `Authors.authorName`.

O Superpoder do CAP: Reutilização com Aspects

Aspects permitem adicionar funcionalidades **transversais** (chaves, auditoria, validade temporal) sem poluir o modelo de domínio, promovendo um código mais limpo e modular.



'cuid' (Canonical UUID Keys)

Adiciona uma chave primária 'ID' do tipo 'UUID'.

```
entity Books : cuid {...}
```



'managed' (Auditoria Automática)

Adiciona campos de auditoria preenchidos automaticamente: 'createdAt', 'createdBy', 'modifiedAt', 'modifiedBy'.

```
entity Orders : managed {...}
```



'temporal' (Dados com Validade Temporal)

Adiciona os campos canônicos 'validFrom' e 'validTo'.

```
entity Contracts : temporal {...}
```

Habilita o suporte nativo do framework para consultas de 'viagem no tempo' e gerenciamento de registros com data de vigência.

Mergulho Profundo: Gerenciando a Dimensão do Tempo

O `aspect temporal` transforma entidades para que elas possam gerenciar dados com validade, permitindo consultas históricas sem lógica customizada complexa.

O Desafio

Como modelar dados que mudam ao longo do tempo (ex: preços de produtos, termos de um contrato)? A abordagem manual é complexa e propensa a erros.

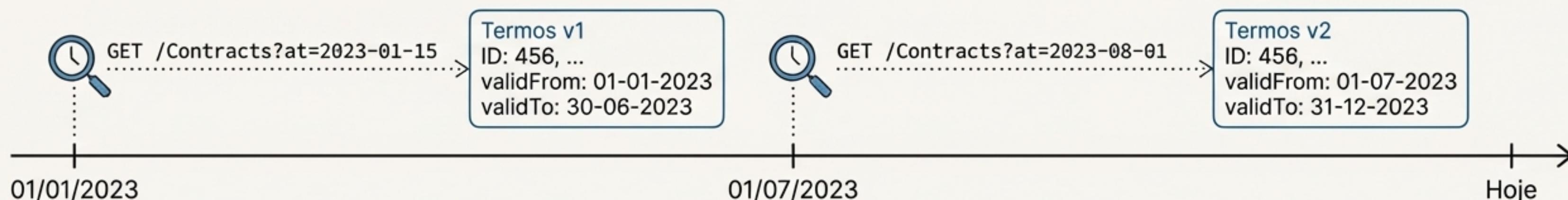
Preços				
ID	Product	Price	valid_from	valid_to
12S	Product X	100.00	01-01-2023	50-06-2023
12S	Product X	110.00	01-07-2023	31-12-2023
12S	Product X	115.00	01-01-2024	99-99-9999
12S	Product X	115.00	01-01-2024	99-99-9999

A Solução CAP

Com uma única linha, }
`entity Contracts :
`temporal` = temporal`, o CAP adiciona os campos
`validFrom` e `validTo` e ativa um suporte em tempo de
execução para:

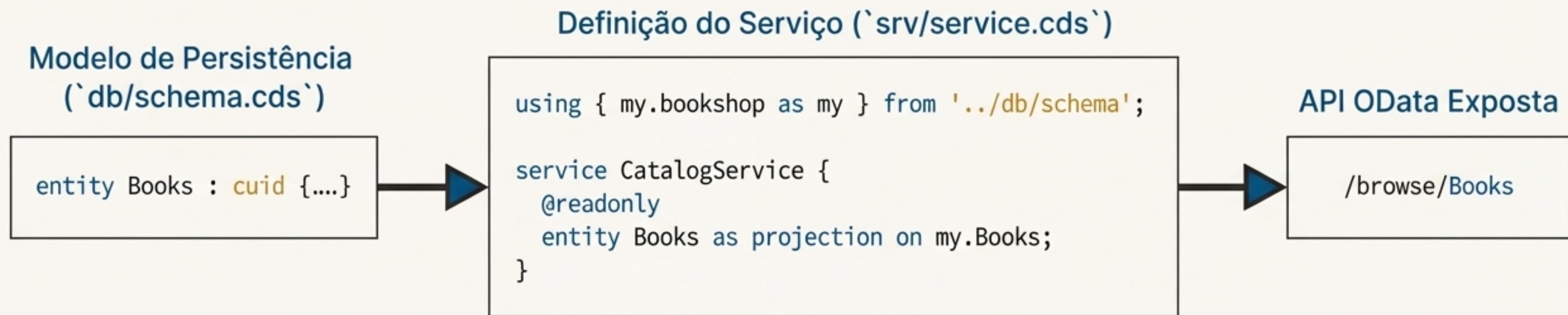
- **Consultas de 'Viagem no Tempo':** Permite consultar o estado dos dados em um ponto específico do passado.
 - **Gerenciamento de Time Slices:** Garante a integridade dos períodos de validade dos registros.

```
cds
entity Contracts : temporal {
    ...
}
```



Passo 2: Expondo Dados com a Camada de Serviço

Com uma simples definição de serviço, seus modelos de dados se transformam em APIs OData robustas, prontas para serem consumidas.



Explicar a separação de interesses: modelos de persistência em `db/` e modelos de consumo em `srv/`.

A Produtividade dos Provedores Genéricos

O CAP gera automaticamente endpoints OData V4 completos, com suporte a consultas complexas, diretamente a partir de suas definições de serviço.

GET /browse/Books?\$select=title,author/name&\$expand=author

```
{  
  "@odata.context": "$metadata#Books(title,author(name))",  
  "value": [  
    {  
      "title": "Wuthering Heights",  
      "author": { "name": "Emily Brontë" }  
    },  
    {  
      "title": "Jane Eyre",  
      "author": { "name": "Charlotte Brontë" }  
    }  
    // ...  
  ]}
```

- \$select (projeção de campos)
- \$filter (filtragem de dados)
- \$expand (expansão de associações)
- \$orderby (ordenação)
- \$top / \$skip (paginação)

Tratando a Complexidade: Dados Localizados de Forma Simples

O modificador `localized` abstrai a complexidade de armazenar e servir textos traduzidos, mantendo o modelo de domínio limpo.

Na Definição (painel esquerdo):

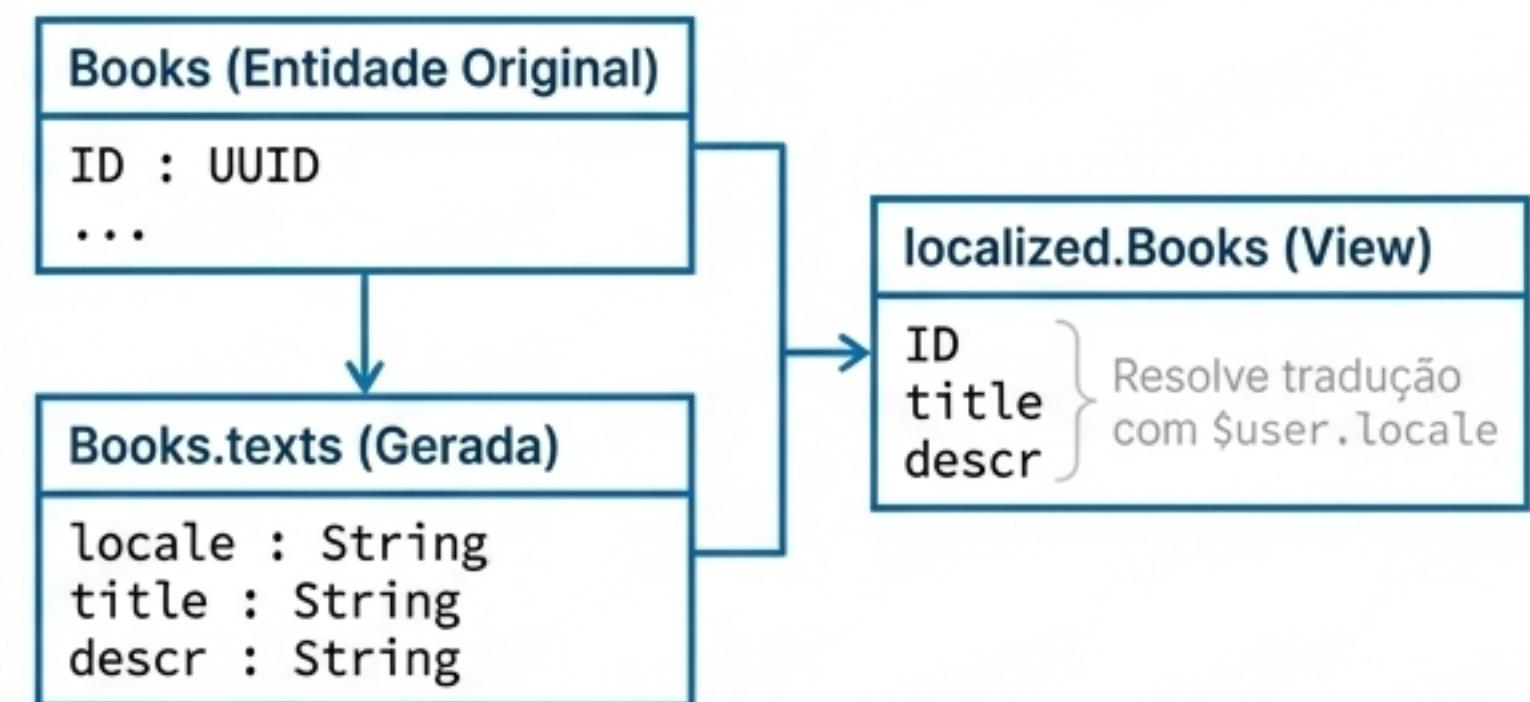
Simplesmente adicione `localized` a um campo de texto.

```
entity Books {  
    key ID : UUID;  
    title : localized String;  
    descr : localized String;  
}
```

Compilador CDS →

Nos Bastidores (painel direito):

O compilador CDS gera uma entidade `Books.texts` separada (ex: `Books.texts`) com uma chave de `locale` e cria uma view que resolve a tradução correta usando a pseudo-variável `\$user.locale`.



Passo 3: Criando Experiências de Usuário com SAP Fiori

Anotações de UI no serviço de backend definem a estrutura e o comportamento da interface, eliminando a necessidade de desenvolver o frontend do zero.

Painel Esquerdo (Código CDS)

```
cds annotate CatalogService.Books with
  @(UI: {
    LineItem: [
      { Value: 'Title' ..... },
      { Value: author.name,
        Label: '{i18n>Author}' }
    ],
    SelectionFields: [
      author_ID,
      stock
    ]
  });

```

Painel Direito (Screenshot da UI)

The screenshot shows a Fiori application interface. On the left, there is a 'Filters' section with two input fields: 'Author' and 'Stock', each with a magnifying glass icon. On the right, there is a 'Books List' section displaying a table with two columns: 'Title' and 'Author'. The table contains two rows of data:

Title	Author
Wuthering Heights	Emily Brontë
Jane Eyre	Charlotte Brontë

Capacitando o Usuário: Suporte a Draft com uma Anotação

Habilite sessões de edição robustas com uma única anotação, permitindo que os usuários salvem seu trabalho em progresso e continuem depois, sem impactar os dados ativos.

Explicação

As alterações são salvas em uma cópia de rascunho até serem ativadas explicitamente.

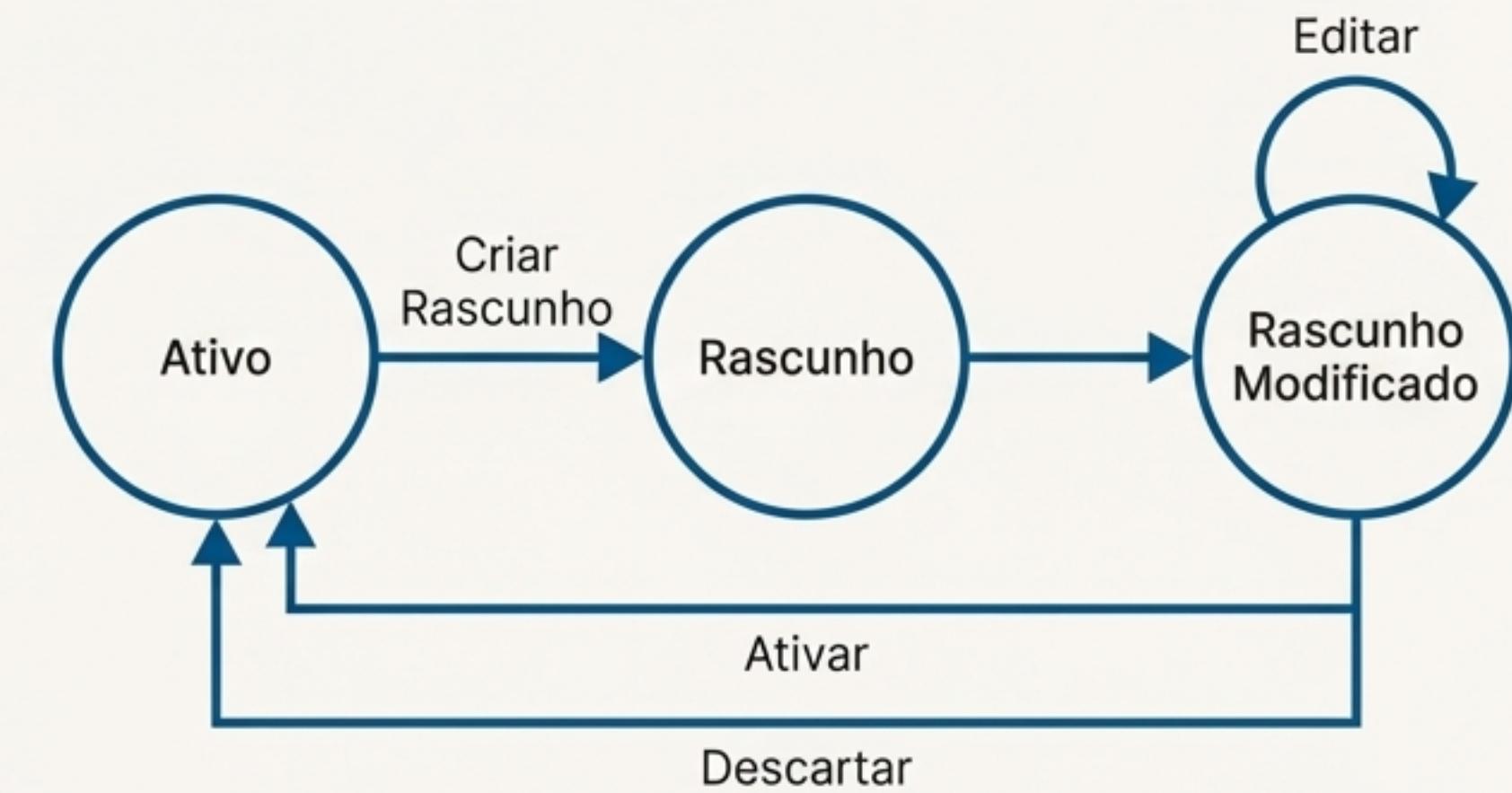
Como Habilitar

```
annotate AdminService.Books with @odata.draft.enabled;
```

O Fluxo

Create Draft → Edit Draft → Activate Draft.

Ações de **PATCH** modificam o rascunho, e uma ação **POST** de ativação move os dados para a entidade ativa.



Passo 4: Fortalecendo a Aplicação — Segurança por Padrão

O CAP adota uma filosofia de 'seguro por padrão', com endpoints protegidos e um modelo de autorização declarativo e poderoso integrado ao CDS.

Autenticação

Por padrão, todos os endpoints do CAP exigem um usuário autenticado. Isso é o comportamento default, garantindo que nada seja exposto acidentalmente.



Autorização Declarativa

Use anotações `@requires` e `@restrict` para definir regras de acesso diretamente no modelo de serviço, mantendo a lógica de segurança junto aos dados que ela protege.

```
entity Orders @restrict: [
    { grant: 'READ', to: 'authenticated-user' },
    { grant: ['CREATE', 'UPDATE'], to: 'Admin' },
]);
```

Acesso de leitura para qualquer usuário logado.

Acesso de escrita restrito ao role 'Admin'.

Fortalecendo a Aplicação — Modelagem para Performance

Construa aplicações rápidas e confiáveis aplicando as melhores práticas de modelagem do CDS para evitar gargalos de performance comuns.

Não Faça

Evite UNION: Geralmente indica uma oportunidade de remodelagem e incorre em penalidades de performance.

Não filtre *depois* do JOIN.

```
// Bad: JOIN completo antes do filtro
view FilteredOrdersJoin as select
  from OrdersHeaders JOIN OrdersItems on ...
  where price > 100;
```

Faça

Prefira modelos que evitem UNIONS.

Filtre na tabela menor *antes* do JOIN.

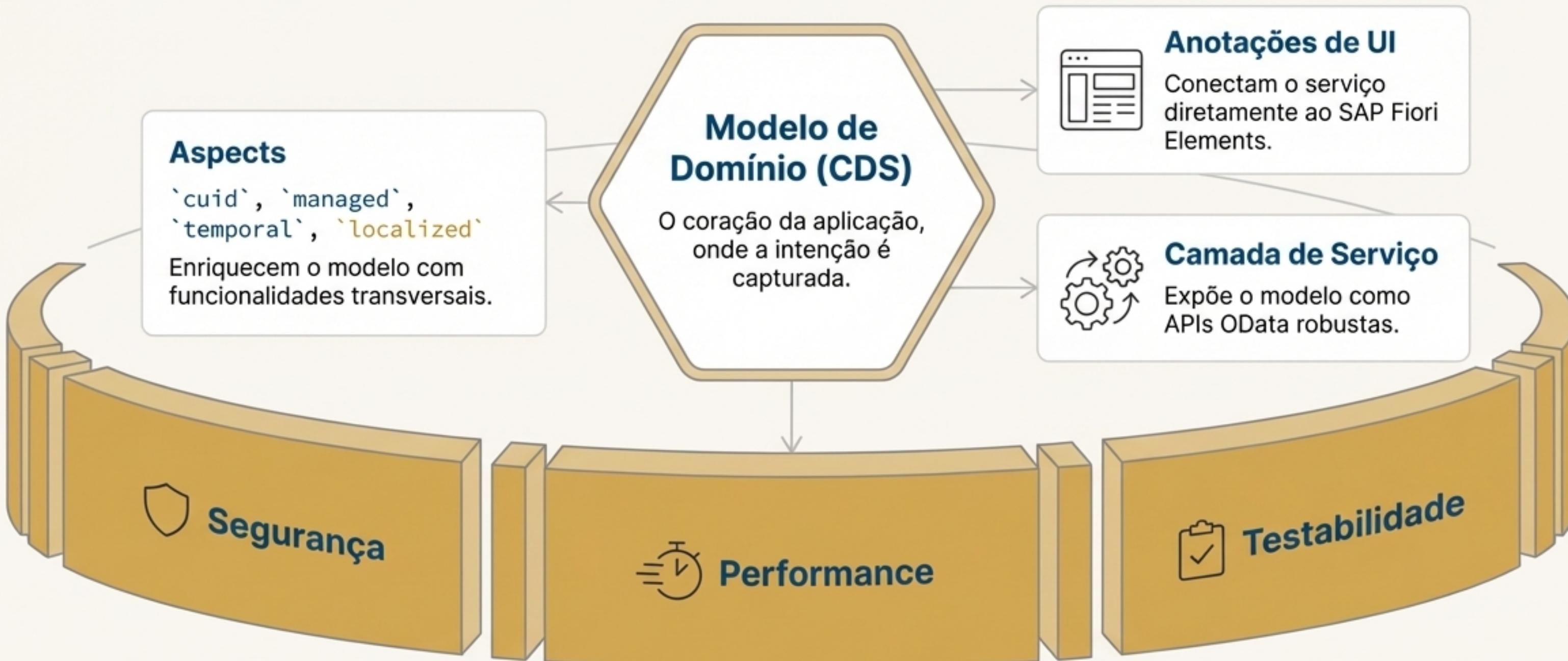
```
// Good: Filtro aplicado antes
// Good: Filtro aplicado antes
view FilteredOrdersAssoc as select from (
  select from OrdersItems {*}
  where price > 100
);
```

Cuidado com Campos Calculados

Cálculos em tempo de leitura (`on read`) impedem o uso de índices. Se possível, pré-calcule os valores na escrita (`on write`) usando um campo `stored`.

A Visão Completa: O Ecossistema Integrado do SAP CAP

O CAP unifica modelagem de domínio, camada de serviço, UIs e qualidades empresariais em um fluxo de trabalho coeso, acelerando o desenvolvimento de ponta a ponta.



Principais Vantagens e Próximos Passos

O SAP CAP acelera o desenvolvimento, promove código limpo e oferece uma base sólida para aplicações empresariais robustas e preparadas para o futuro.

Vantagens Resumidas

-  **Desenvolvimento Orientado a Modelos**
Foco na intenção de negócio.
-  **Produtividade Massiva**
Geração automática de APIs e suporte a UI.
-  **Elegância e Reutilização**
Separação de interesses clara com Aspects ([temporal](#), [managed](#), etc.).
-  **Ecossistema Integrado**
Sinergia perfeita do backend ao frontend.
-  **Qualidade Empresarial**
Segurança, performance e outros recursos essenciais incorporados por padrão.

Próximos Passos (Recursos para Exploração)

-  **Leia a Documentação**
A fonte definitiva de conhecimento sobre o CAP.
[cap.cloud.sap/docs \(capire\)](http://cap.cloud.sap/docs)
-  **Explore os Exemplos**
Veja implementações reais e aprenda com código funcional.
github.com/sap-samples/cloud-cap-samples
-  **Comece um Projeto**
Inicie um novo projeto com [cds init](#) e explore os aspectos [temporal](#) e [localized](#).