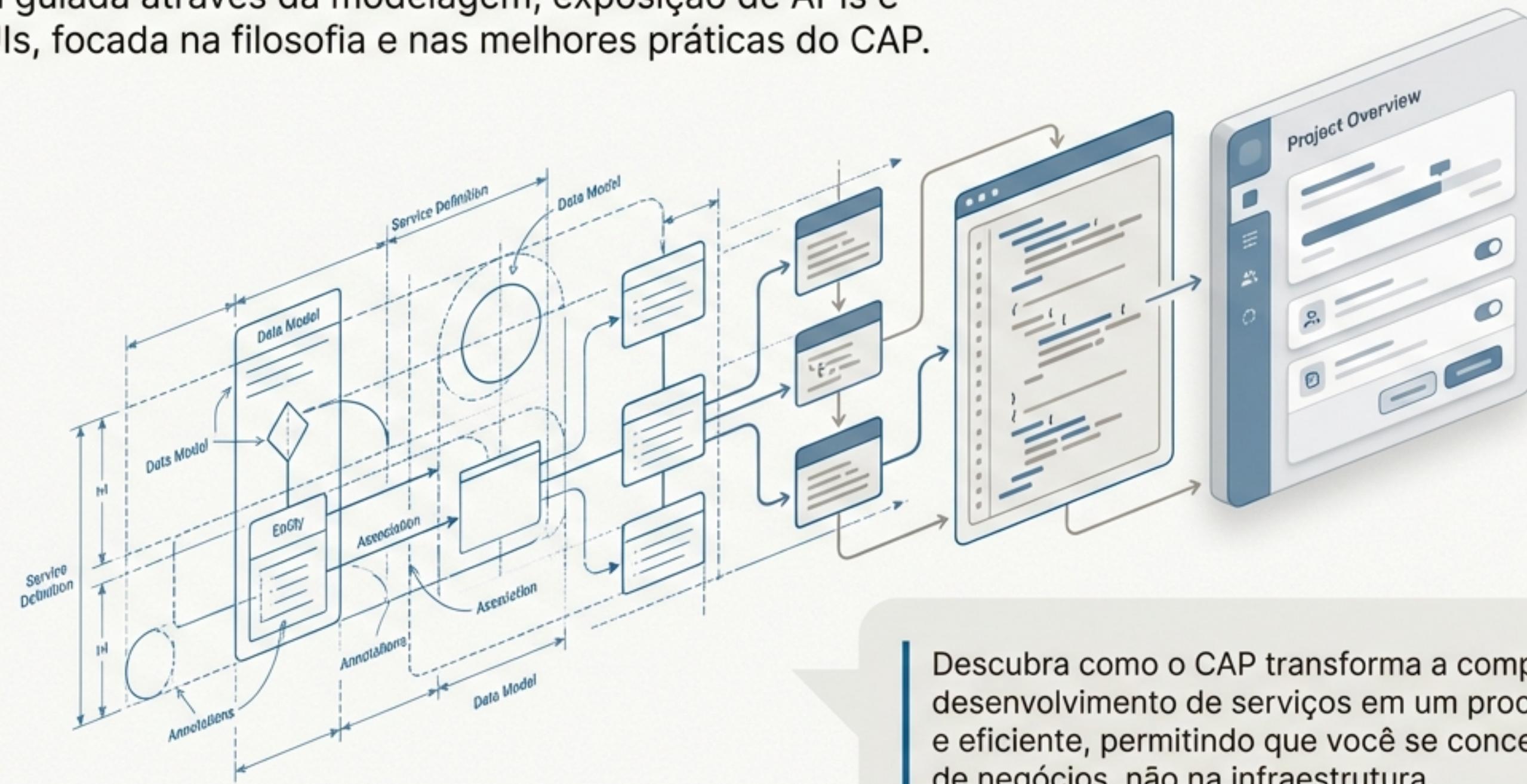


Construindo Serviços com CAP: Da Intenção à Interface

Uma jornada guiada através da modelagem, exposição de APIs e criação de UIs, focada na filosofia e nas melhores práticas do CAP.



Descubra como o CAP transforma a complexidade do desenvolvimento de serviços em um processo declarativo e eficiente, permitindo que você se concentre na lógica de negócios, não na infraestrutura.

A Filosofia CAP: Capture a Intenção, Não o 'Como'

O CAP foi projetado para focar na modelagem conceitual. A prioridade é capturar a intenção do domínio de negócio de forma clara e concisa. O framework se encarrega de fornecer implementações genéricas e otimizadas, liberando o desenvolvedor da necessidade de escrever código repetitivo (boilerplate).

A Intenção

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

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



O Resultado



Persistência Otimizada: Tabelas de banco de dados com chaves primárias UUID e campos de auditoria (`createdAt`, `createdBy`, etc.).



APIs OData Ricas: Endpoints prontos com suporte a `'\$select'`, `'\$filter'`, `'\$expand'`.



Suporte a Localização: Geração automática de entidades `'.texts'` para dados multilíngues.



Segurança Integrada: Ganchos para autorização e validação.

A Jornada do Desenvolvedor: Uma Construção em 5 Etapas



Cada etapa se baseia na anterior, transformando um modelo de dados simples em uma aplicação completa e pronta para produção.

Etapa 1: O Alicerce - Modelagem de Domínio

Os modelos de domínio são a base para a persistência e para as definições de serviço. Eles capturam os aspectos estáticos e de dados do seu problema.



Requisito: "Queremos uma livraria onde usuários possam navegar por Livros e Autores, e ver o Gênero de cada livro."

Modelo CDS Resultante

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

entity Books : cuid {
    title : String;
    descr : String;
    genre : Genre;
    author : Association to Authors;
}

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

type Genre : String enum {
    Mystery; Fiction; Drama;
}
```

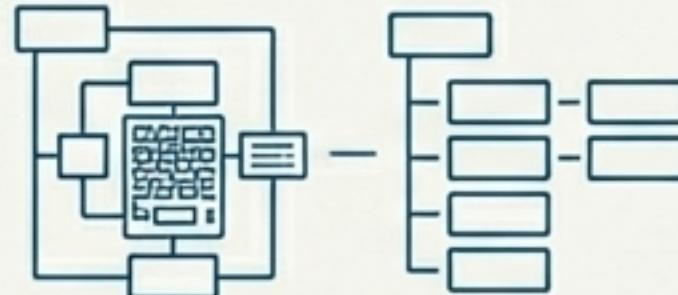


Dica de Mestre: Use os Aspectos Comuns!

‘cuid’: Adiciona uma chave primária canônica ‘ID : UUID’, que é preenchida automaticamente.

‘managed’: Adiciona quatro campos de auditoria (‘createdAt’, ‘createdBy’, ‘modifiedAt’, ‘modifiedBy’) que são gerenciados automaticamente pelo runtime do CAP.

Boas Práticas do Alicerce: Mantenha Simples e Organizado



Prefira Modelos Planos (Flat)

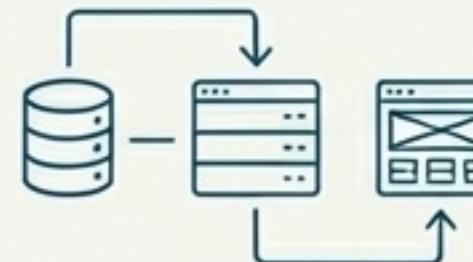
Evite estruturas de tipos profundamente aninhadas. Estruturas planas são mais fáceis de entender, consumir e integrar com outras tecnologias.

Don't

```
entity Contacts {  
    personData : PersonDetails;  
}
```

Do

```
entity Contacts {  
    firstname: String;  
    lastname: String;  
}
```



Separação de Interesses

Mantenha seu modelo de domínio principal limpo. Use arquivos `*.cds` separados para anotações de UI, autorização e outros aspectos secundários. Isso melhora a legibilidade e a manutenção.

```
db/  
  schema.cds      (Modelo de Domínio)  
srv/  
  service.cds     (Definição de Serviço)  
  app/annotations.cds (Anotações de UI)
```



Convenções de Nomes

Adote convenções claras para consistência.

- **Entidades no Plural:** `Books`, `Authors`.
- **Tipos no Singular:** `Genre`.
- **Nomes de Entidades/Tipos em Maiúscula:** `Books`.
- **Nomes de Elementos em Minúscula:** `title`.
- **Prefira Nomes Concisos:** `Authors.name` em vez de `Authors.authorName`.

Etapa 2: O Núcleo - Definindo o Contrato do Serviço

Um serviço CAP expõe entidades do seu modelo de domínio. Ele pode expor as entidades diretamente ou através de 'projeções', que funcionam como views, permitindo customizar a API sem alterar o modelo de dados subjacente.

1. Importando o Modelo de Domínio

```
ng { my.bookshop as db } from '../db/schema';
```



2. Definindo o Serviço e Expondo Entidades:

```
service CatalogService {  
    // Exposição direta da entidade Authors  
    entity Authors as projection on db.Authors;  
  
    // Projeção com um subconjunto de campos e anotações  
    entity Books as projection on db.Books {  
        @readonly  
        ID,  
        title,  
        author,  
        stock  
    };  
};
```



O Poder das Projeções

Use projeções para criar APIs específicas para diferentes casos de uso (leitura, administração, etc.) sem duplicar seu modelo de dados. Você pode adicionar anotações (@readonly), remover campos ou até mesmo achatar associações diretamente na definição do serviço.

Etapa 3: A Porta de Entrada - Exposição Automática via OData

Uma vez que um serviço é definido em CDS, o runtime do CAP o expõe automaticamente como um serviço OData V4 completo. Não é necessário escrever código para implementar os handlers básicos de CRUD ou as complexas opções de consulta.



Recursos OData Disponíveis 'Out-of-the-Box'		
Opção de Consulta	Descrição	Status
\$select	Seleciona um subconjunto de propriedades.	Suportado
\$filter	Filtra a coleção de recursos (equivalente a `WHERE`).	Suportado
\$expand	Inclui entidades associadas no resultado.	Suportado
\$orderby	Ordena os resultados.	Suportado
\$top / \$skip	Implementa a paginação dos resultados.	Suportado
\$count	Retorna o número de entidades no conjunto.	Suportado

Exemplo de Requisição: `GET /catalog/Books?\$select=title,stock&\$filter=stock > 10`

Enriquecendo a API com Anotações OData

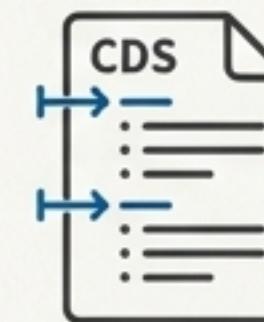
Anotações são metadados que adicionam semântica ao seu serviço. Elas descrevem como os dados devem ser interpretados e apresentados, controlando o comportamento de clientes genéricos. O CAP permite definir essas anotações diretamente no CDS.

Exemplo Prático: Adicionando anotações para definir um título (@Common.Label) e informações de cabeçalho (@UI.HeaderInfo) a uma entidade.

```
// Em um arquivo app/annotations.cds
using { CatalogService } from '../srv/service';

annotate CatalogService.Books with @(
    Common.Label: 'Livro',
    UI.HeaderInfo: {
        TypeName      : 'Livro',
        TypeNamePlural: 'Livros',
        Title         : { Value : title }
    }
);
```

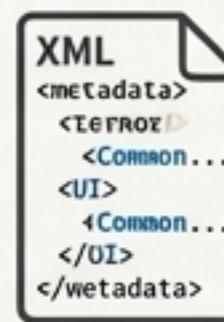
Como Funciona:



O modelo CDS com
anotações...



...é compilado pelo CAP
para um arquivo EDMX
(metadata.xml)...



...que descreve os
termos do vocabulário
OData ([Common](#), [UI](#),
etc.) para os clientes.



Dica de Mestre: Prefira '@title' e '@description'!

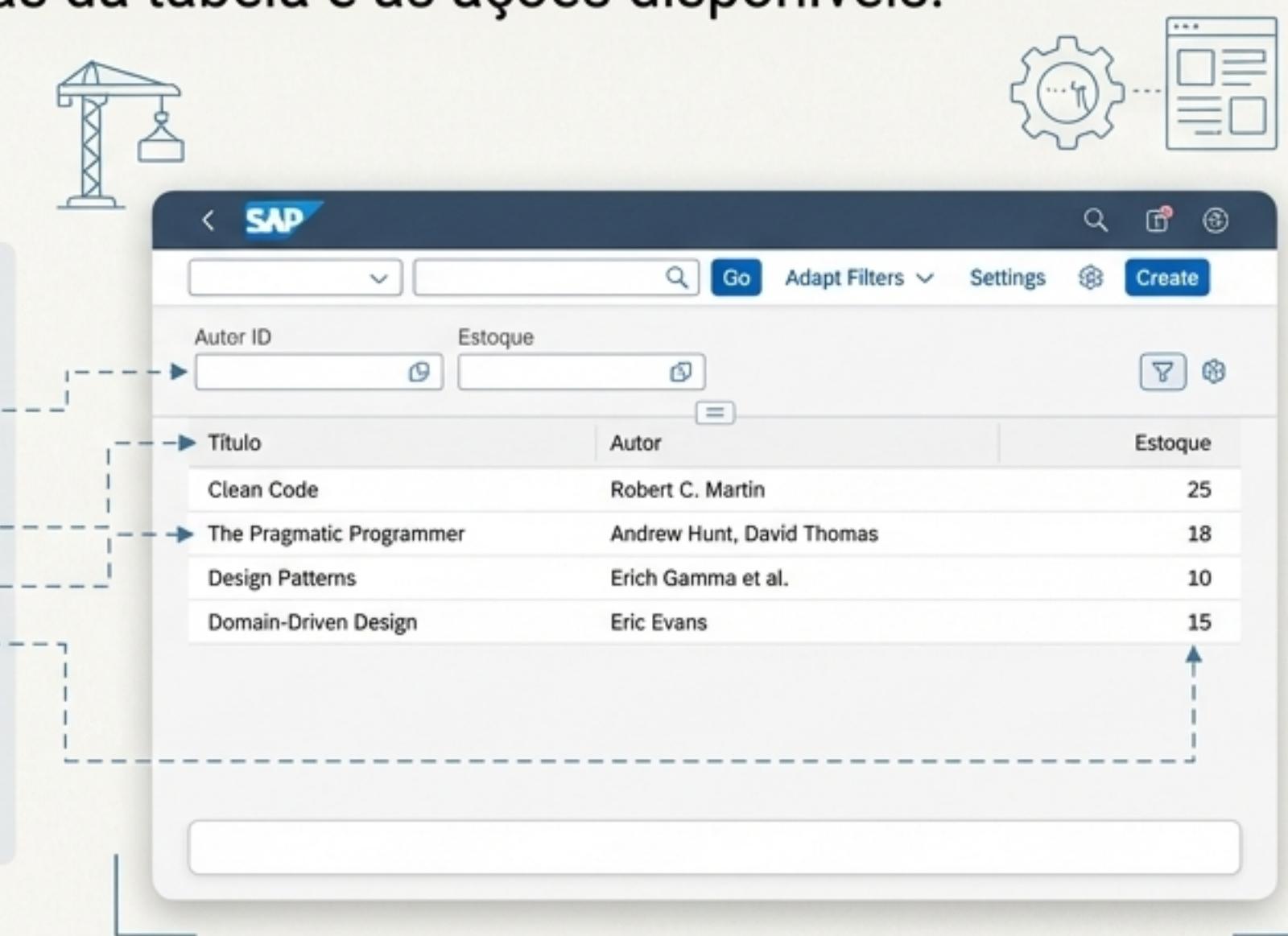
Para semânticas comuns de rótulo e descrição, use as anotações `@title: 'Meu Título'` e `@description: 'Minha Descrição'`. O CAP as mapeia automaticamente para `@Common.Label` e `@Core.Description` no OData, mantendo seu modelo agnóstico de protocolo.

Etapa 4: A Fachada - Servindo UIs com SAP Fiori Elements

As **aplicações SAP Fiori Elements** são clientes genéricos que constroem a interface do usuário dinamicamente com base nos metadados e anotações OData do seu serviço. As anotações definem o layout, os campos de busca, as colunas da tabela e as ações disponíveis.

Exemplo de Anotação de UI:

```
annotate CatalogService.Books with @(
    UI: {
        SelectionFields: [ author_ID, stock ],
        LineItem: [
            { Value: title, Label: 'Título' },
            { Value: author.name, Label: 'Autor' },
            { Value: stock, Label: 'Estoque' }
        ]
    }
);
```



Uma Fachada Robusta: Habilitando Suporte a Rascunhos (Draft)

O suporte a 'drafts' permite que os usuários editem dados em uma cópia de rascunho, salvando o progresso sem afetar os dados ativos. As alterações só são aplicadas na entidade principal após uma ação explícita de 'Salvar'. O CAP e o Fiori Elements fornecem suporte completo para este padrão.

Como Habilitar:
Adicione uma única anotação à sua entidade de serviço.

```
`cds\  
annotate CatalogService.Books with @odata.draft.enabled;
```

A Coreografia do Draft



POST (vazio): Cria um Rascunho.
O servidor retorna um rascunho com
'IsActiveEntity = false'.



PATCH: Edita o Rascunho.
O usuário modifica os campos. As
alterações são salvas no rascunho.



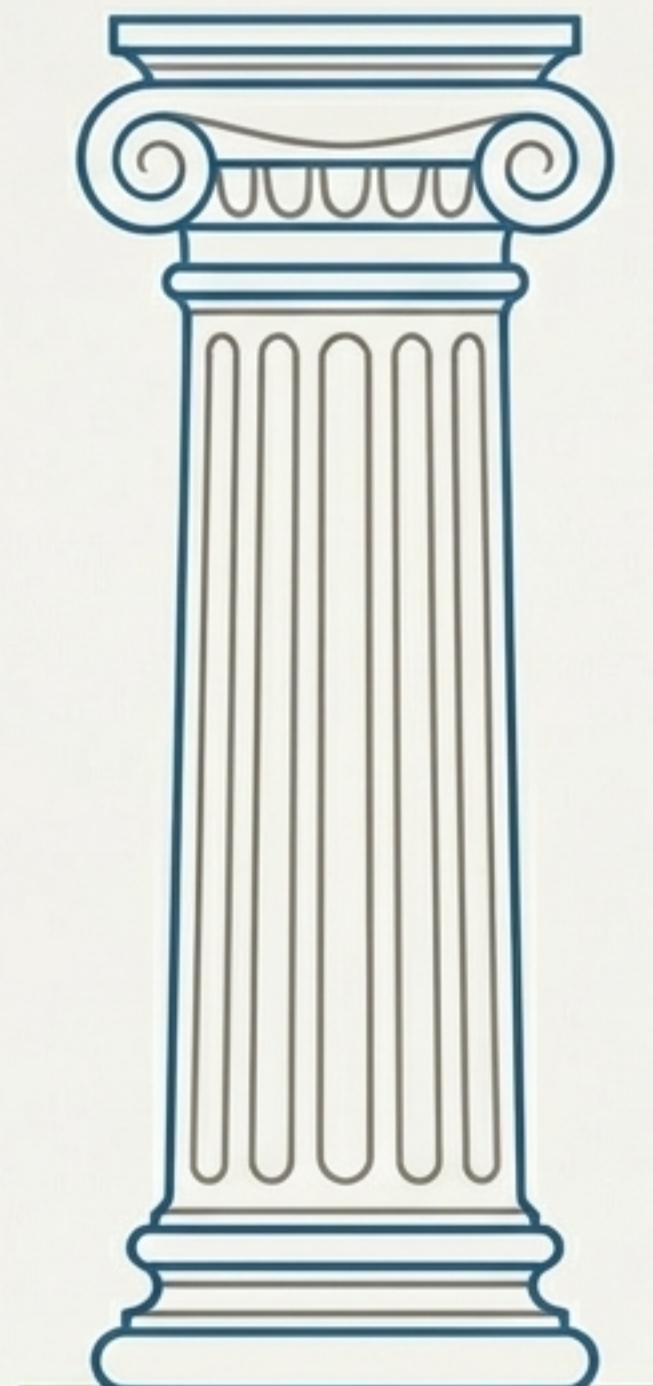
POST (DraftActivate): Ativa o Rascunho.
As alterações são validadas e movidas para
a entidade ativa ('IsActiveEntity = true').

Benefícios Chave:

- Permite sessões de edição longas e interrompidas.
- Validações podem ser executadas no rascunho antes de salvar.
- Evita bloqueios (locking) na entidade ativa durante a edição.

Etapa 5: Os Pilares - Garantindo Qualidades Essenciais

Um serviço não é definido apenas por suas funcionalidades, mas também por sua robustez, segurança e capacidade de atender a um público global. O CAP fornece mecanismos integrados para garantir essas qualidades essenciais.



Pilares a Serem Explorados:

Segurança por Design:
Protegendo seus dados e APIs.

Performance na Modelagem:
Escrevendo modelos eficientes desde o início.

Suporte a Dados Localizados:
Atendendo a usuários em diferentes idiomas.



Pilar 1: Segurança por Design

Filosofia

O CAP é projetado para ser seguro por padrão ('Secure by Default').

- ✓ Endpoints são autenticados por padrão quando o XSUAA está configurado.
- ✓ A comunicação é criptografada via HTTPS/TLS no nível da plataforma (SAP BTP).
- ✓ O framework é imune a ataques de injeção de SQL via CQN, que utiliza prepared statements.



Atenção

Nunca inclua roles técnicas como `cds.Subscriber` ou `mtcallback` em roles denegócio. Garanta que as roles de plataforma sejam gerenciadas separadamente.

Exemplo Prático

Autorização Declarativa

A autorização é definida diretamente no modelo CDS usando anotações. Isso mantém a lógica de segurança junto ao modelo de dados, tornando-a fácil de entender e auditar.

```
`cds\  
// Acesso de leitura para todos os usuários autenticados  
// Acesso completo apenas para a role 'content-maintainer'  
@restrict: [  
    { grant: 'READ', to: 'authenticated-user' },  
    { grant: ['CREATE', 'UPDATE', 'DELETE'], to: 'content-maintainer' }  
]  
entity Books { ... }
```



Pilar 2: Performance na Modelagem

A performance começa no modelo. Decisões de modelagem têm um impacto significativo nas consultas ao banco de dados.



Evite 'UNION' em views.

Problema: `UNION` pode causar lentidão e complexidade. É um sinal de que o polimorfismo pode ser modelado de forma mais eficiente.
Solução: Refatore para usar entidades semânticas comuns ou associações.



Filtre e Ordene na Tabela Certa.

Problema: Fazer um `JOIN` grande e depois filtrar/ordenar o resultado materializa dados desnecessários.
Solução: Modele com associações e use projeções. Filtre na tabela de "itens" primeiro, antes de "juntar" com o cabeçalho.



Evite Campos Calculados "on read".

Problema: Cálculos como concatenação de strings ou fórmulas em tempo de leitura impedem o uso de índices e causam 'full table scans'.
Solução (apresentada em ordem de preferência):
1. **Preferencial:** Calcule na UI. 2. **Alternativa:** Pré-calcule no momento da escrita ('on write') usando um campo 'stored'.



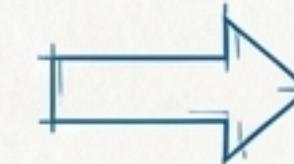
Pilar 3: Dados Localizados de Forma Simples

Para campos que precisam de tradução, basta adicionar o modificador `localized`. O CAP cuida de toda a complexidade por trás dos panos.

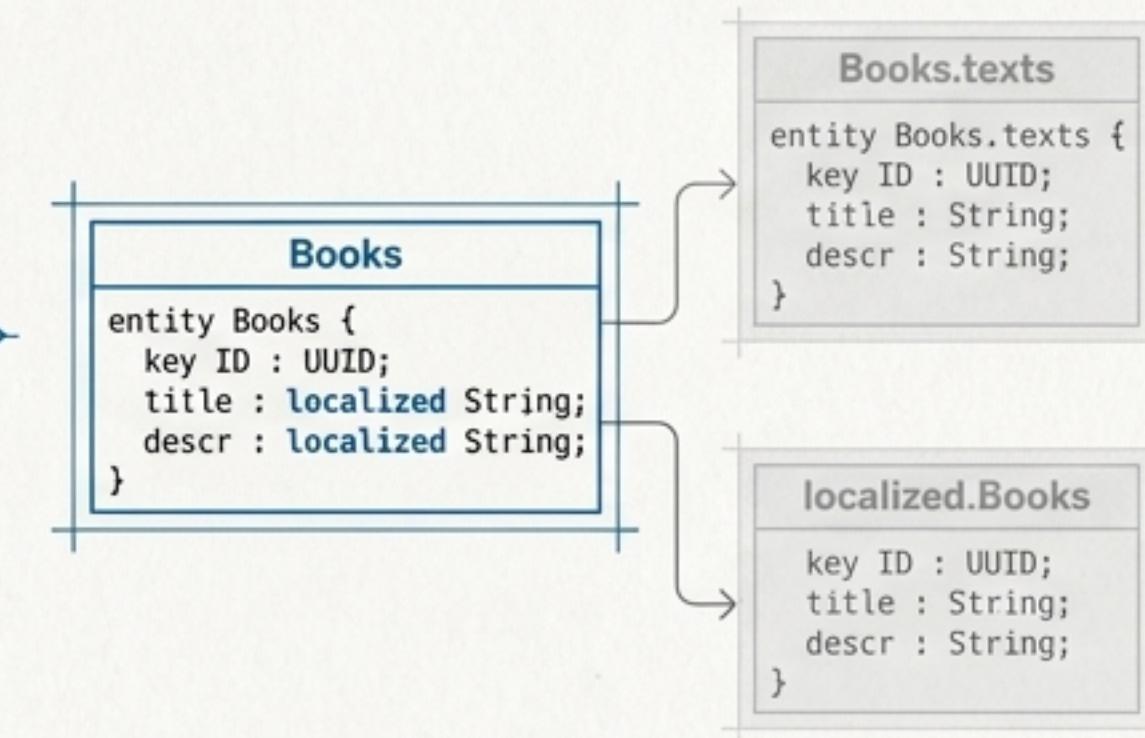
Como Funciona

1. No seu Modelo (schema.cds):

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



2. O que o CAP Gera (Automaticamente):



Entidade de textos:

```
entity Books.texts {  
    key locale : sap.common.Locale;  
    key ID : UUID;  
    title : String;  
    descr : String;  
}
```

View de tradução:

Uma view `localized.Books` une a entidade base com a tabela de textos, usando a variável `$user.locale`.

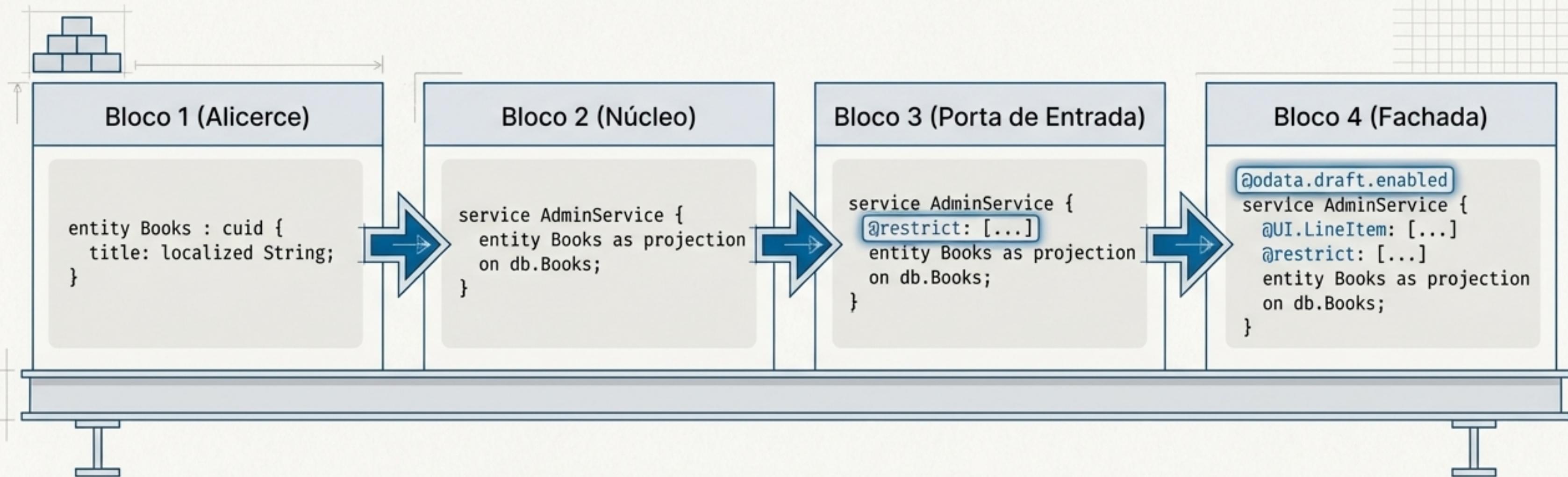
Fornecendo Dados Iniciais:

Crie dois arquivos CSV: `Books.csv` para os dados no idioma padrão e `Books_texts.csv` para as traduções.

Books.csv
ID,title,descr
...

Books_texts.csv
locale,ID,title,descr
...

A Jornada Completa: Do Modelo à API Segura e Pronta para UI



Seguindo a jornada do desenvolvedor, vimos como o CAP traduz um modelo de domínio simples e semântico em um serviço empresarial completo. Ao focar na **intenção**, você constrói aplicações **mais rápidas, consistentes e fáceis de manter**, enquanto o framework cuida da complexidade da implementação.

CAP: Menos código, mais domínio.