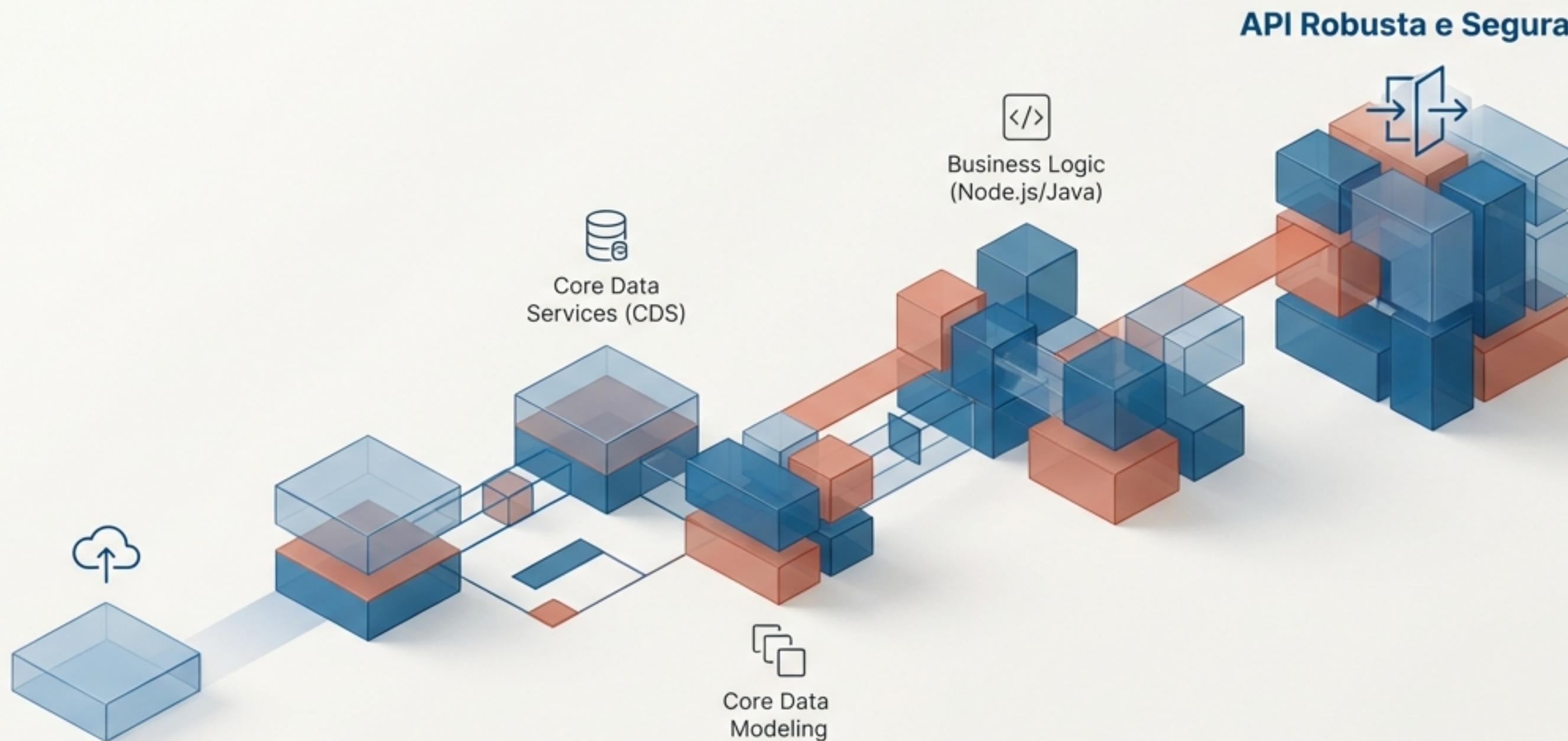


A Jornada do Desenvolvedor CAP

Do Modelo Conceitual à API Robusta e Segura



A Filosofia Central: Foco na Intenção

“Capture a Intenção, não o Como!”

O CAP foca na modelagem conceitual do domínio de negócio. O desenvolvedor declara **o que** a aplicação deve fazer, e o framework, através de provedores genéricos, cuida da implementação otimizada, seja no banco de dados ou nos serviços.

```
// Um modelo conciso que captura intenções complexas
using { cuid, managed } from '@sap/cds/common';

entity Books : cuid, managed {
    title : localized String;           // Intenção: Texto traduzível
    author : Association to Authors;   // Intenção: Relacionamento gerenciado
}
```

Chave UUID
única e canônica

Dados de auditoria
(criado/modificado por/em)

Suporte a múltiplos idiomas
de forma transparente

Passo 1: Construindo a Fundação com CDS

O modelo de domínio em CDS é a espinha dorsal da aplicação, definindo entidades, tipos e seus relacionamentos.

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

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

entity Authors : cuid {
    name : String;
    // Associação de volta (backlink) gerenciada pelo CAP
    books : Association to many Books on books.author = $self;
}
```

Boas Práticas

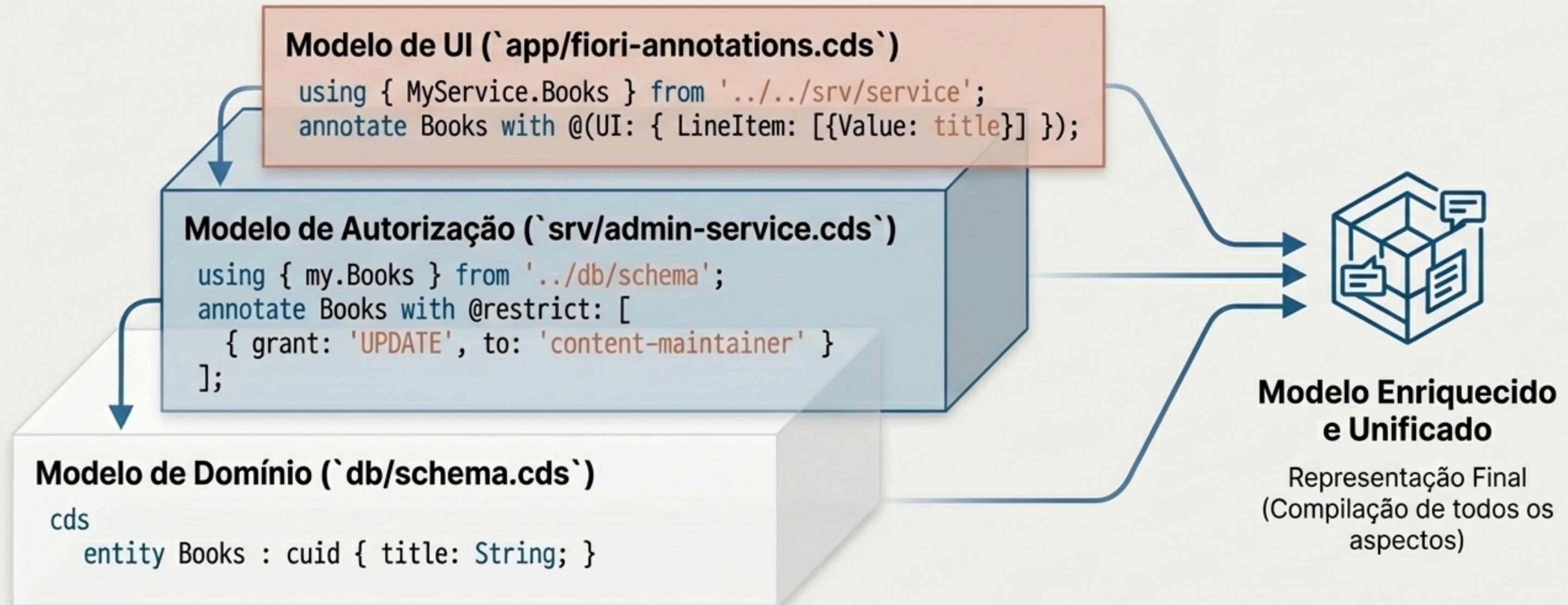
Chaves Canônicas: Prefira chaves primárias técnicas e simples. Use `cuid` para chaves `UUID` canônicas.

Convenções de Nomes: Use plural para nomes de entidades (ex: `Books`) e singular para tipos.

Nomes Concisos: Evite redundância. Prefira `Authors.name` em vez de `Authors.authorName`.

O Poder dos Aspectos: Separando as Preocupações

Os Aspectos do CDS permitem decompor definições em múltiplos arquivos, aplicando características secundárias ao modelo de domínio principal sem 'poluí-lo'.

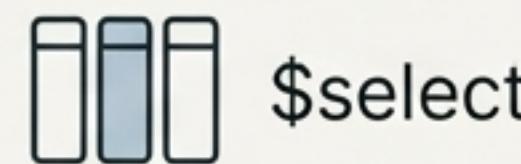


Passo 2: Publicando a API com OData

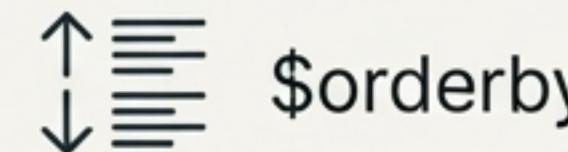
Com uma simples definição de serviço em CDS, o CAP expõe automaticamente uma API OData V4 completa e pronta para consumo.



Funcionalidades OData Inclusas:



\$select



\$orderby



\$filter



\$top / \$skip



\$expand



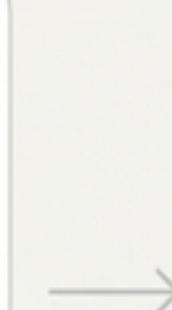
\$count

Enriquecendo APIs com Anotações OData

Anotações OData adicionam uma camada de metadados semânticos à API, descrevendo como os dados devem ser renderizados e usados. Elas são a ponte entre o backend e UIs inteligentes como SAP Fiori.

.cds

```
annotate CatalogService.Books with @(
    UI: {
        LineItem: [ // Define as colunas de uma tabela na UI
            {Value: title},
            {Value: author.name, Label:'Autor'},
            {Value: price, Label:'Preço'}
        ]
    }
);
```

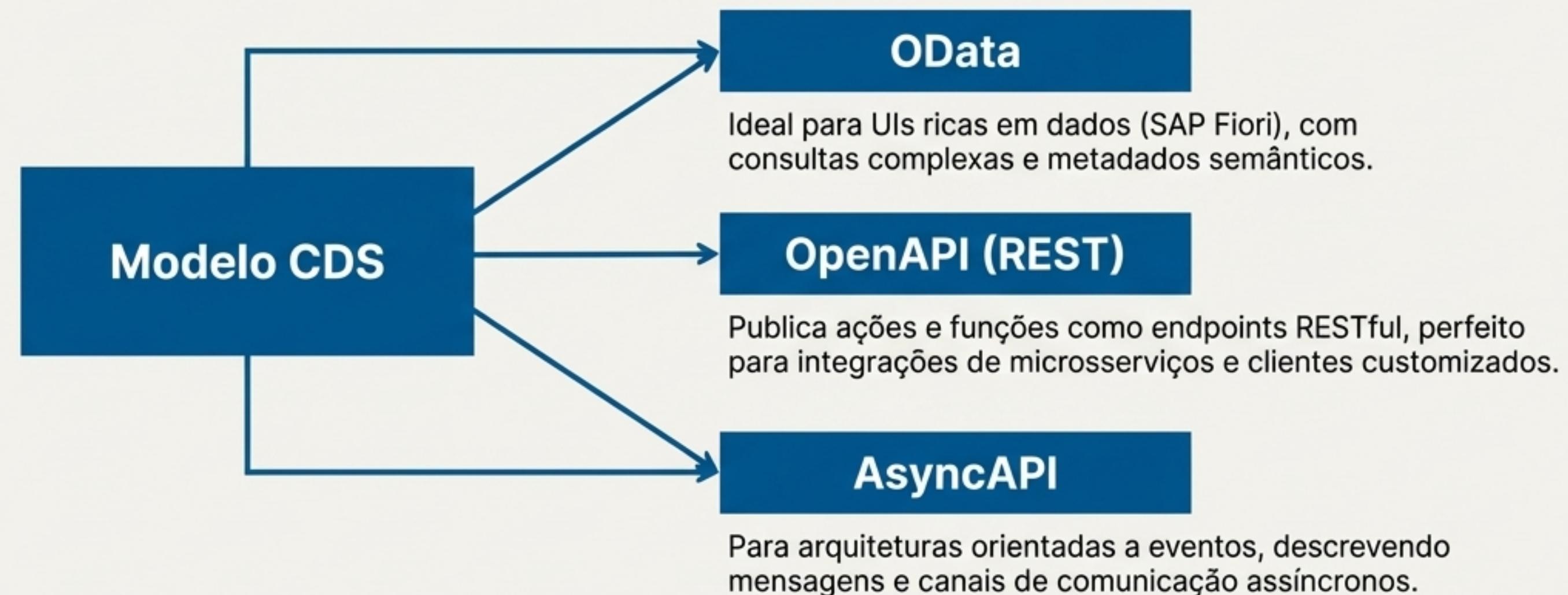


Title	Autor	Preço
Wuthering Heights	Emily Brontë	12.99
Jane Eyre	Charlotte Brontë	14.50
Pride and Prejudice	Jane Austen	10.95

Esta anotação instrui um cliente SAP Fiori a renderizar uma tabela com as colunas 'title', 'author.name' e 'price', sem a necessidade de escrever uma única linha de código de UI.

Além do OData: Flexibilidade para OpenAPI e AsyncAPI

O mesmo modelo CDS que define seus dados e lógica de negócios pode ser publicado em múltiplos formatos de API para atender a diferentes casos de uso.



A publicação para OpenAPI e AsyncAPI é habilitada através de configurações e ferramentas específicas do CAP, alavancando o mesmo modelo de serviço.

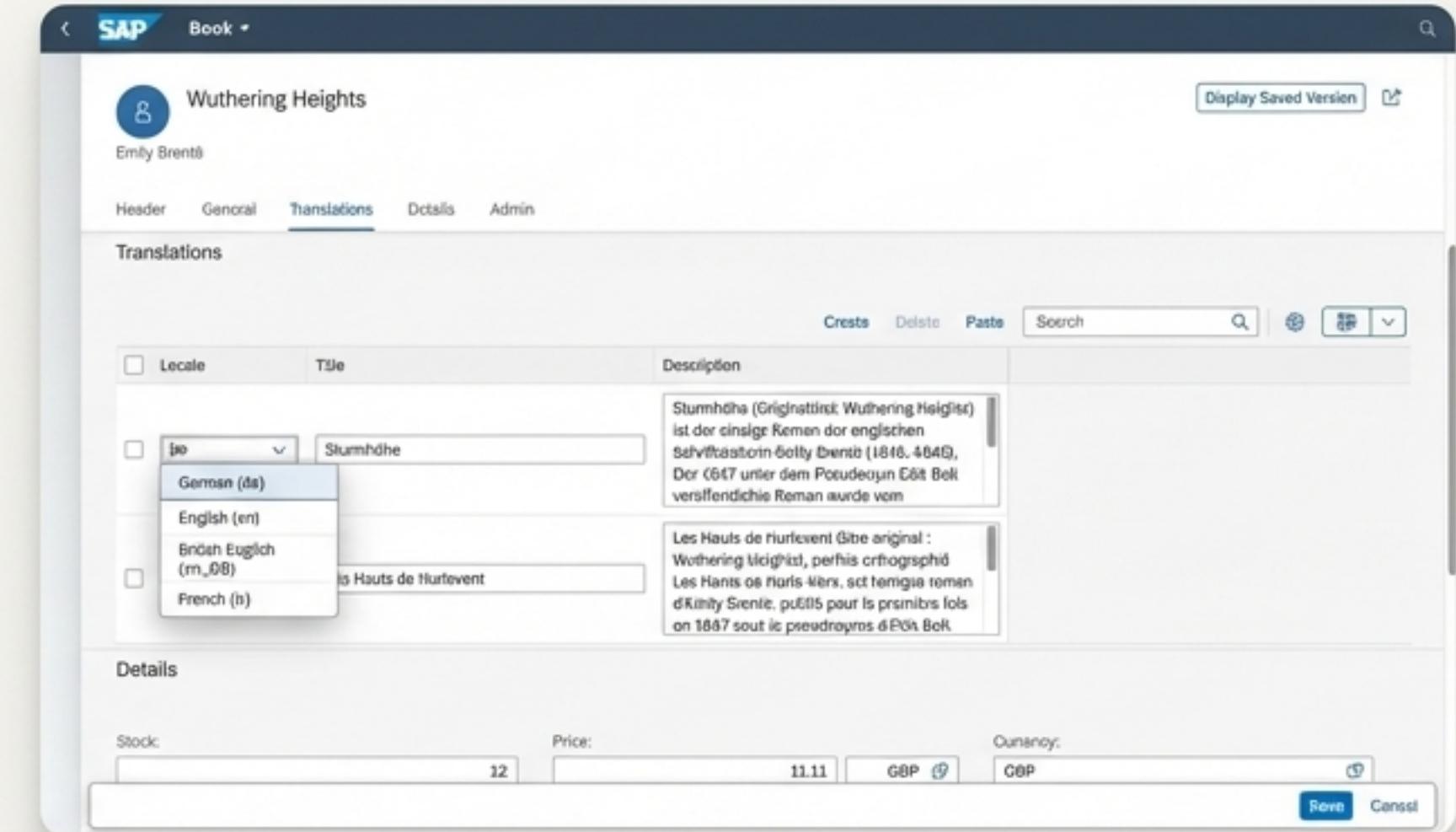
Da API à Aplicação: Suporte Nativo a SAP Fiori Elements

As anotações OData alimentam diretamente as aplicações SAP Fiori Elements, gerando UIs ricas e consistentes com esforço mínimo de desenvolvimento.

Funcionalidade em Destaque: Suporte a Draft

Permite que os usuários salvem alterações em rascunho no servidor, sem afetar os dados ativos. Podem continuar a edição mais tarde, em qualquer dispositivo.

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



Passo 3: Criando um Ecossistema Conectado

Consumindo APIs externas de forma nativa.

1

Importar a Definição da API

Use a CLI do CAP para importar a especificação do serviço remoto (EDMX) e gerar um modelo CDS.

```
cds import API_BUSINESS_PARTNER.  
edmx --as cds
```

2

Configurar a Conexão

Declare o serviço externo no `package.json` para que o CAP gerencie a conectividade.

```
"cds": {  
  "requires": {  
    "API_BUSINESS_PARTNER": {  
      "kind": "odata",  
      ...  
    }  
  }  
}
```

3

Consumir em Código

Acesse o serviço remoto com a mesma API de consulta (CQN) usada para o banco de dados local.

```
const bupa = await cds.connect.to  
('API_BUSINESS_PARTNER');  
  
const result = await  
bupa.run(SELECT.from(bupa.entities  
.A_BusinessPartner));
```

Acelerando o Desenvolvimento: Mocking e Testes Locais

Mocking Automático

Durante o desenvolvimento local (`cds watch`), o CAP automaticamente simula (`mock`) os serviços externos definidos no `package.json`.

```
[cds] - serving RiskService { at: '/service/risk', impl: './srv/risk-service.js' }
[cds] - mocked 'API_BUSINESS_PARTNER' at '/service/api-business-partner'
```

Dados de Teste

É possível fornecer dados de teste para os serviços mockados através de arquivos `.csv` no diretório `srv/external/data`.

Testes Integrados

A biblioteca `@cap-js/cds-test` oferece um framework robusto para testes unitários e de integração.

```
const { GET, expect } = cds.test(__dirname+'/..');

it('deve retornar a lista de livros', async () => {
  const { data } = await GET('/browse/Books');
  expect(data.value).to.have.lengthOf.at.least(1);
});
```

Passo 4: Qualidade e Alcance Global com Dados Localizados

O Desafio

Aplicações **globais** precisam exibir dados (como descrições de produtos) no idioma do usuário. Gerenciar tabelas de tradução manualmente é complexo e propenso a erros.

A Solução Elegante do CAP

```
cds entity Books {  
    key ID : UUID;  
    title : localized String; // <- É só isso!  
    descr : localized String;  
}
```

Como Funciona ('Nos Bastidores'):



O compilador CDS cria uma entidade **Books.texts** com a chave **locale**.

É gerada uma view chamada **localized.Books**.

Essa view usa **coalesce** e a variável **\$user.locale** para buscar a tradução correta.

O runtime do CAP usa essa view automaticamente.

Passo 5: Construindo uma API Segura por Design



Princípio Fundamental: Secure by Default.

Por padrão, os endpoints do CAP são protegidos e exigem autenticação assim que a integração com um provedor de identidade (como o XSUAA) é configurada.



Autorização Declarativa no Modelo.

As regras são parte do design da API, definidas diretamente no modelo CDS, tornando-as explícitas e fáceis de auditar.



Integração Nativa.

O CAP integra-se com serviços do SAP BTP como XSUAA e Connectivity Service, abstraindo a complexidade.

```
annotate Books with @(restrict: [
  { grant: 'READ', to: 'authenticated-user' },
  { grant: 'CREATE', to: 'content-maintainer' },
  { grant: 'UPDATE', to: 'content-maintainer' },
  { grant: 'DELETE', to: 'admin' },
]);
```

Blindando sua API: Mitigação de Ameaças Comuns

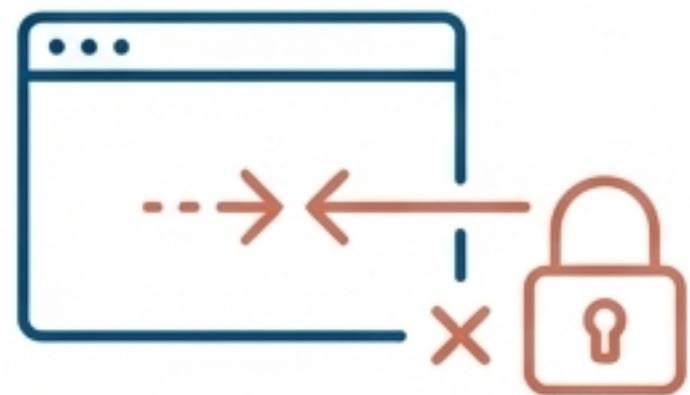
O CAP oferece proteções intrínsecas contra uma série de vetores de ataque comuns, reduzindo a superfície de ataque da sua aplicação.



SQL Injection:

A API de consulta (CQL/CQN) é imune por padrão, pois é convertida em *prepared statements*.

Atenção: Valide inputs do usuário se estiver construindo a estrutura da query dinamicamente.



CSRF (Cross-Site Request Forgery):

Aplicações com UI são protegidas pelo SAP Application Router, que gerencia a proteção de token CSRF (x-csrf-token) automaticamente.



Negação de Serviço (DoS):

O runtime implementa paginação automática para limitar o consumo de memória. O número de requisições em um \$batch pode ser limitado via configuração (cds.odata.batch_limit).

Passo 6: Otimizando para Performance em Escala

A performance no CAP começa na modelagem. Decisões de design em CDS impactam diretamente a eficiência das consultas no banco de dados.



Evite `UNION` em Views:

Causa penalidades de performance e aumenta a complexidade. Para polimorfismo, prefira modelar com associações a uma entidade base.



Cuidado com Campos Calculados:

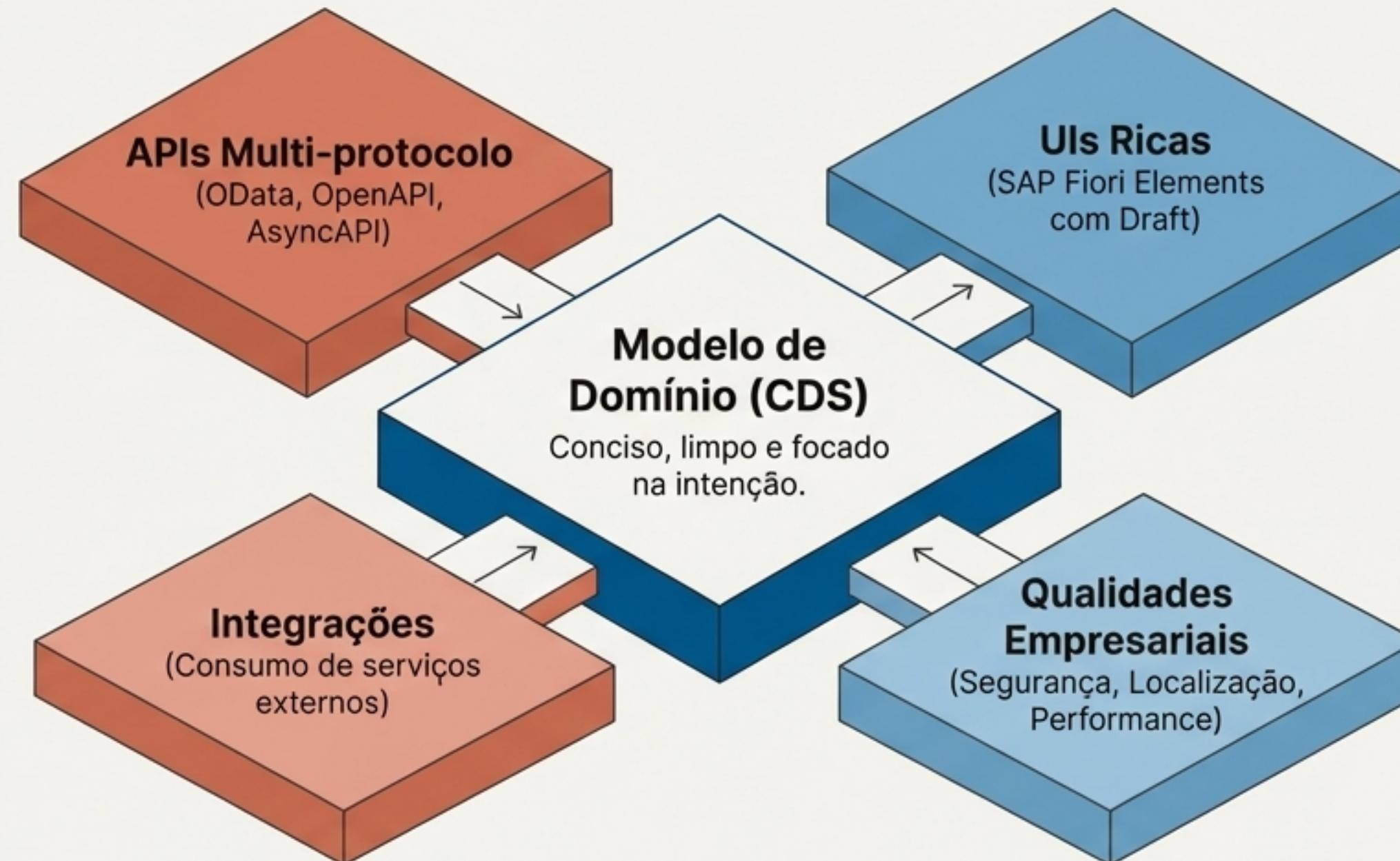
Cálculos em tempo de leitura (`on read`) impedem o uso de índices e podem causar `full table scans`. Prefira calcular na UI ou pré-calcular na escrita (`stored`).



Filtre e Ordene Cedo:

Em views com JOINs, aplique filtros (`where`) e ordenação (`order by`) na tabela mais específica *antes* de fazer o JOIN, para que o banco de dados trabalhe com um conjunto de dados menor.

A Jornada Completa: Do Conceito à Realidade Empresarial



O SAP Cloud Application Programming Model transforma um modelo de domínio conceitual em uma **aplicação corporativa completa**. Ele acelera a jornada de desenvolvimento, garantindo que a aplicação final seja robusta, segura e escalável, do início ao fim.