

# **# Construindo Aplicações Robustas com SAP CAP**

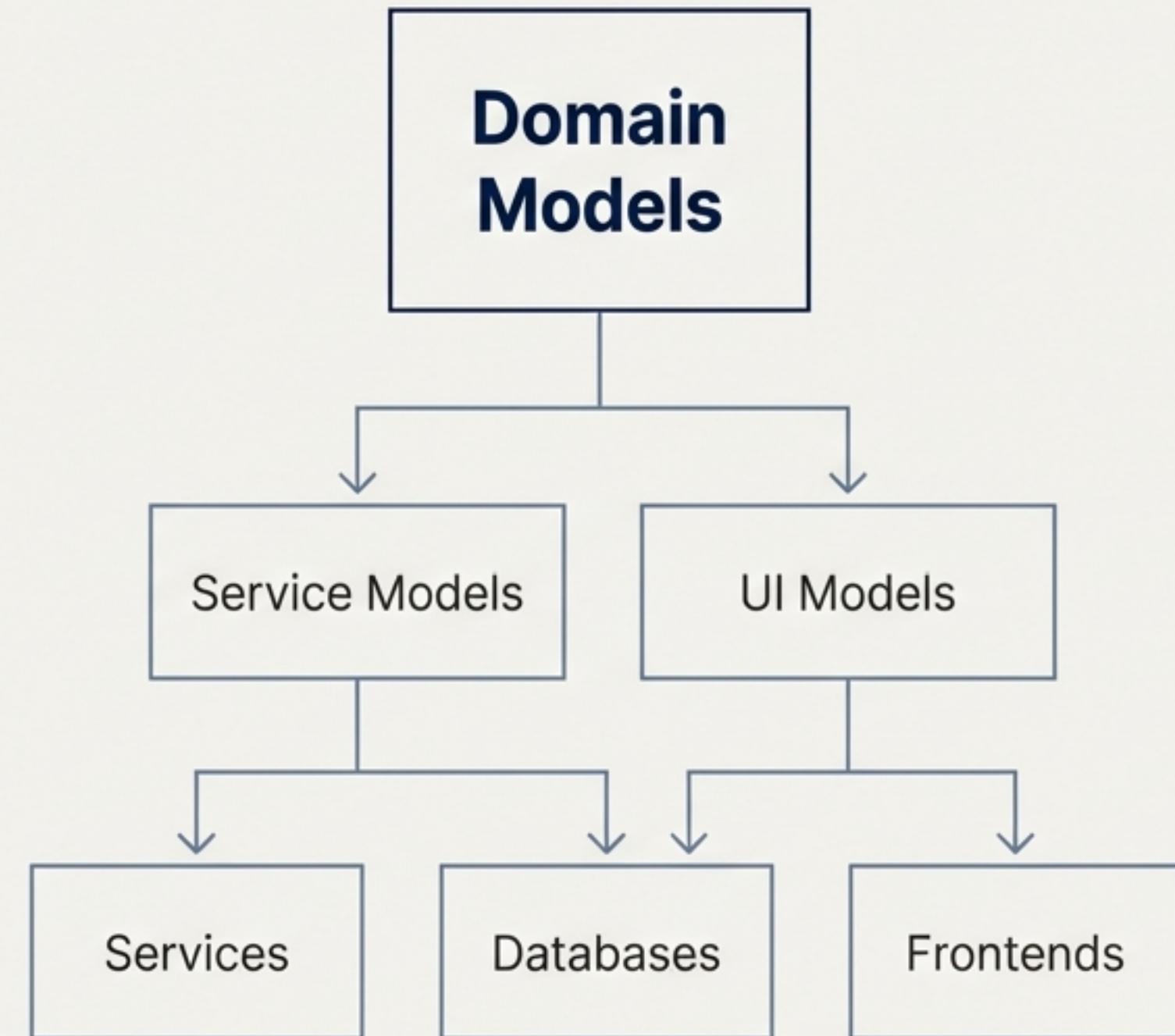
### Dos Fundamentos à Excelência Operacional

> “O CDS foca na modelagem conceitual: queremos capturar a **intenção**, não as implementações imperativas — ou seja: **O Que, não o Como.**”

\*Fonte: CAP Cookbook 2 - *Domain Modeling*

## Princípios Fundamentais:

- **Modelos Concisos e Compreensíveis:** O código é a documentação, promovendo a colaboração entre desenvolvedores e especialistas de domínio.
- **Implementações Genéricas Otimizadas:** Deixe que o framework cuide das tarefas repetitivas e complexas.
- **Foco no Domínio Central:** A energia do projeto é concentrada no domínio principal do negócio.



# O Alicerce: A Arte da Modelagem de Domínio

O ponto de partida de qualquer aplicação robusta é um modelo de domínio limpo e expressivo, definido em CDS (Core Data Services).

## Exemplo Básico: Uma Livraria

```
// db/schema.cds
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;
}
```

\*Fonte: CAP Cookbook 2 - *Domain Modeling*

## Regras de Ouro da Modelagem:

- 1. Mantenha a Simplicidade (KISS):** Prefira modelos planos e evite abstrações excessivas.
- 2. Nomes Concisos e Consistentes:** Entidades no plural (ex: `Authors`), tipos no singular (ex: `Genre`). Elementos em minúsculas.
- 3. Separação de Interesses:** Mantenha o modelo de domínio limpo. Anotações de UI, autorização, etc., devem ficar em arquivos separados.

# Acelerando o Desenvolvimento com Aspectos Reutilizáveis

Em vez de reescrever código repetitivo, use os aspectos pré-definidos do @sap/cds/common para declarar sua intenção.

## `cuid`: Chaves Primárias Canônicas

O que você escreve:

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



É expandido para:

```
entity Books {  
    key ID : UUID;  
    ...  
}
```

É expandido para: }

Garante chaves primárias UUID únicas e universais, preenchidas automaticamente pelo runtime.

## `managed`: Dados de Auditoria

O que você escreve:

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



É expandido para:

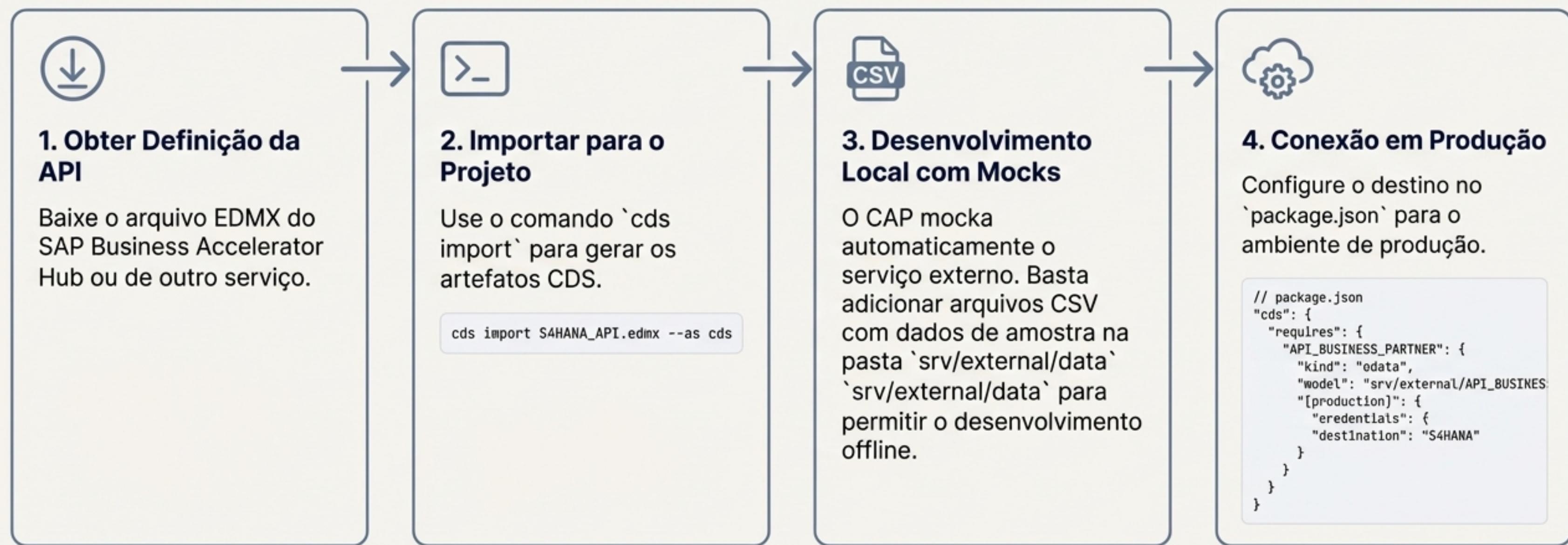
```
entity Orders {  
    createdAt : Timestamp @cds.on.insert: $now;  
    createdBy : User @cds.on.insert: $user;  
    modifiedAt : Timestamp @cds.on.update: $now;  
    modifiedBy : User @cds.on.update: $user;  
    ...  
}
```

Adiciona automaticamente campos de criação e modificação, gerenciados pelo runtime.

# Dando Vida ao Modelo: Consumindo Serviços Externos

Nenhuma aplicação é uma ilha. O CAP simplifica a integração com serviços remotos.

## O Fluxo de Consumo:



# Expondo APIs OData de Alta Qualidade com Facilidade

Exponha seus modelos de domínio como serviços OData V4 robustos com o mínimo de esforço.

## Funcionalidades OData Suportadas Nativamente:



**Consultas Ricas:** `'\$select`, `\$expand`, `\$filter`, `\$orderby`, `\$search`, `\$apply` (agregação).



**Modificação de Dados:** `POST`, `PATCH`, `PUT`, `DELETE` com suporte a ETags para controle de concorrência.



**Mapeamento Inteligente de Tipos:** Tipos CDS são mapeados automaticamente para os tipos OData EDM corretos.



**Suporte a Rascunho (Draft):** Essencial para aplicações Fiori, habilitado com uma única anotação.

## Definição de Serviço em Duas Linhas:

```
// srv/cat-service.cds
using { my.bookshop as my } from '../db/schema';

service CatalogService {
    entity Books as projection on my.Books;
    entity Authors as projection on my.Authors;
}
```

**Anotações para Potencializar:** Use anotações para enriquecer os metadados do serviço, guiando clientes como as UIs Fiori.

```
annotate CatalogService.Books with @UI.LineItem: [...];
```

# Potencializando UIs com SAP Fiori Elements

Construa UIs Fiori ricas e consistentes através de **metadados, não de código de UI imperativo.**

The screenshot shows a Fiori application for managing books. At the top, there's a header with the SAP logo and a back arrow. Below it, the title 'Wuthering Heights' by 'Emily Brontë'. The navigation bar includes 'Header', 'General', 'Translations' (which is underlined), 'Details', and 'Admin'. The main content area is titled 'Translations' and contains a table with columns 'Locale', 'Title', and 'Description'. There are three rows: one for German (De) with the value 'Sturmhöhe', one for French (Fr) with the value 'Les Hauts de Hurlevent', and one for English (En) with the value 'Wuthering Heights'. A tooltip for the German row provides a detailed description of the book. At the bottom, there are sections for 'Stock', 'Price', and 'Currency', with 'Save' and 'Cancel' buttons.

```
// app/browse/fiori-service.cds
using { CatalogService } from '../../../../../srv/cat-service';

// Anota a entidade Books do serviço
annotate CatalogService.Books with @(
    UI: {
        LineItem: [ // Define as colunas da lista
            {Value: title},
            {Value: author.name, Label:'Autor'},
            {Value: price},
            {Value: currency.symbol, Label: ' '}
        ],
        SelectionFields: [ author_ID, price, currency_code ]
    }
);
```

**O Papel das Anotações:** As anotações em seu serviço CDS descrevem a semântica dos dados, permitindo que o Fiori Elements renderize as páginas e os controles corretos.

**Prática Recomendada (Separação de Interesses):** Mantenha suas anotações de UI na pasta `app/`, preservando a clareza do seu serviço (`srv/`).

**Supporte a Rascunho (Draft):** Habilite sessões de edição complexas e seguras com uma única anotação no serviço:

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

\*Fonte: CAP Cookbook 7 - Serving SAP Fiori UIs

# Construindo para um Mundo Global: Dados Localizados

A internacionalização (i18n) está integrada diretamente ao modelo de dados, não é uma reflexão tardia.

## O Modificador `localized`

Basta adicionar a palavra-chave `localized` aos campos de texto que precisam de tradução.

```
// db/schema.cds
entity Books {
    key ID : UUID;
    title : localized String; // ✨ Intenção declarada aqui
    descr : localized String; // ✨ E aqui
}
```

## Como Funciona:

O CAP gera automaticamente uma entidade `.texts` para armazenar as traduções e uma view que une os dados, selecionando o idioma correto com base no `'\$user.locale'`.

## Carregando Dados Iniciais:



`Books.csv`: Contém os dados no idioma padrão.



`Books\_texts.csv`: Contém as traduções para os outros idiomas.



```
// Books_texts.csv
ID,locale,title,descr
201,de,Sturmhöhe,Sturmhöhe (Originaltitel: Wuthering Heights) ist der einzige...
201,fr,Les Hauts de Hurlevent,Les Hauts de Hurlevent (titre original : Wutheri...
```

# Segurança por Design, Não por Acaso

O CAP incorpora princípios de segurança desde o início, protegendo sua aplicação por padrão.

## Pilares da Segurança em CAP:



### Comunicação Segura

O SAP BTP garante canais de comunicação criptografados (HTTPS/TLS) por padrão para todos os endpoints.



### Autenticação Integrada

Integração nativa com o serviço XSUAA do BTP. Endpoints são protegidos por padrão; você deve declarar explicitamente os que são públicos. O comando `cds add xsuua` configura a instância.



### Autorização Declarativa

Defina quem pode fazer o quê diretamente no seu modelo de serviço usando anotações legíveis.

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



### Isolamento de Dados (Multi-Tenancy)

Ao configurar multitenancy, o runtime do CAP isola automaticamente os dados por tenant em contêineres HDI dedicados, garantindo que um assinante não possa ver os dados de outro.

\*Fonte: CAP Cookbook 12.3 - Security Aspects

# Um Bom Modelo Também é um Modelo Performático

Um modelo de dados correto não é suficiente. Ele precisa ser performático para escalar.

**Filtros e Joins:** Filtre na tabela certa antes\* do JOIN.



## Ruim

Fazer o JOIN de duas tabelas grandes e depois filtrar o resultado massivo.

```
// View que faz JOIN e depois filtra
view FilteredOrdersJoin as select from OrdersHeaders
    JOIN OrdersItems on ...
    where price > 100;
```



## Bom

Usar associações e projeções. O filtro é aplicado na fonte de dados antes da união, resultando em uma consulta muito mais eficiente.

```
// A projeção permite o pushdown do filtro
entity FilteredOrdersAssoc as projection on OrdersHeaders;
// O filtro é aplicado eficientemente no $expand
GET /OrderItemsViewAssoc?$expand=Items($filter=price gt 100)
```

**Campos Calculados:** Cálculos em tempo de leitura ('on read') impedem o uso de índices e podem causar *full table scans*.



## Ruim

Calcular uma categoria em uma view a cada leitura.

```
// Causa full table scan se filtrado/ordenado
case when quantity > 100 then 'Medium'
      else 'Small' end as category
```



## Bom

Pré-calcular o valor na escrita ('on write') usando um campo 'stored'. O cálculo é feito uma vez, a leitura é rápida.

```
extend my.OrdersItems with {
    category: String = case ... end stored;
}
```

# Garantindo a Qualidade com Testes Integrados

A biblioteca `@cap-js/cds-test` fornece utilitários para escrever testes eficazes e de ponta a ponta para suas aplicações CAP, com suporte para Jest e Mocha.

```
// test/services.test.js
const cds = require('@sap/cds')
const { expect } = require('chai')

describe('Testes do Serviço de Catálogo', () => {
  // Inicia um servidor CAP para os testes
  const { GET, POST } = cds.test.in(__dirname, '...')

  // Testa a API HTTP
  it('Deve retornar os livros publicados', async () => {
    const { data } = await GET('/browse/Books')
    expect(data.value).to.not.be.empty
    expect(data.value[0]).to.have.property('title')
  })
})
```

Testando suas APIs de forma simples e direta.

Testa a API HTTP

```
// Testa a API programática (lógica de serviço)
it('Deve reduzir o estoque ao submeter um pedido', async () => {
  const srv = await cds.connect.to('CatalogService')
  const { Books } = srv.entities
  const bookBefore = await SELECT.one.from(Books, 201)

  // Simula a ação de submeter um pedido
  await POST('/browse/submitOrder', { book: 201, quantity: 5 })

  const bookAfter = await SELECT.one.from(Books, 201)
  expect(bookAfter.stock).to.equal(bookBefore.stock - 5)
})
```

Testa a API programática (lógica de serviço)

# A Jornada com CAP: Resumo da Filosofia

O SAP Cloud Application Programming Model em resumo:



## Model-Driven:

O modelo de domínio CDS é a fonte única da verdade, impulsionando o banco de dados, os serviços e as UIs.



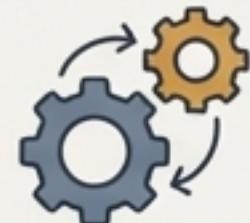
## Foco na Intenção:

Declare “o que” você quer (managed, localized, cuid), e deixe o framework otimizar o “como”.



## Serviços de Alta Qualidade por Padrão:

Exponha e consuma APIs robustas (OData) com segurança, resiliência e funcionalidades ricas integradas.



## Ecossistema Coeso:

Da modelagem aos testes, uma experiência de desenvolvimento unificada e produtiva.



## Aberto e Extensível:

Construído sobre padrões abertos (CDS, OData) e facilmente extensível com lógica de negócios personalizada em Node.js ou Java.

# Continue sua Jornada com CAP

Explore os recursos oficiais para aprofundar seu conhecimento e começar a construir.

## Recursos Essenciais:

**Documentação Oficial (capire):** A fonte definitiva para guias, conceitos e referências.  
[cap.cloud.sap](https://capire.cloud.sap)

**Tutoriais e Exemplos:** Projetos de exemplo práticos que cobrem uma ampla gama de  
[github.com/sap-samples/cloud-cap-samples](https://github.com/sap-samples/cloud-cap-samples)

**SAP Fiori Tools:** Explore as extensões para VS Code e Business Application Studio para acelerar o desenvolvimento de UIs.