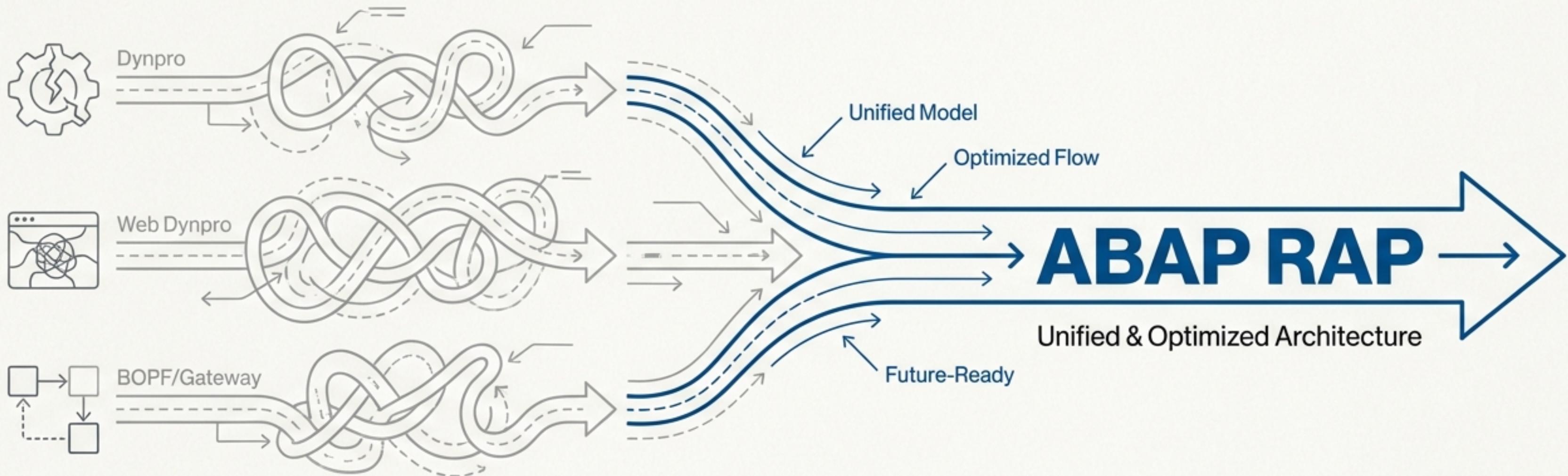


# De Mestre de Obras a Arquiteto de Software

## Dominando o ABAP RESTful Application Programming Model (RAP)

# A Evolução Inevitável: Por que o RAP é o Padrão

O ABAP RESTful Application Programming Model (RAP) não é apenas uma nova ferramenta; é a culminação de décadas de evolução. Ele substitui modelos anteriores fragmentados por uma arquitetura unificada e opinativa, projetada para o futuro do SAP.



## Padrão Estratégico

Arquitetura definitiva para **SAP S/4HANA** (On-Premise e Cloud) e **ABAP Cloud**.



## Nativo para HANA

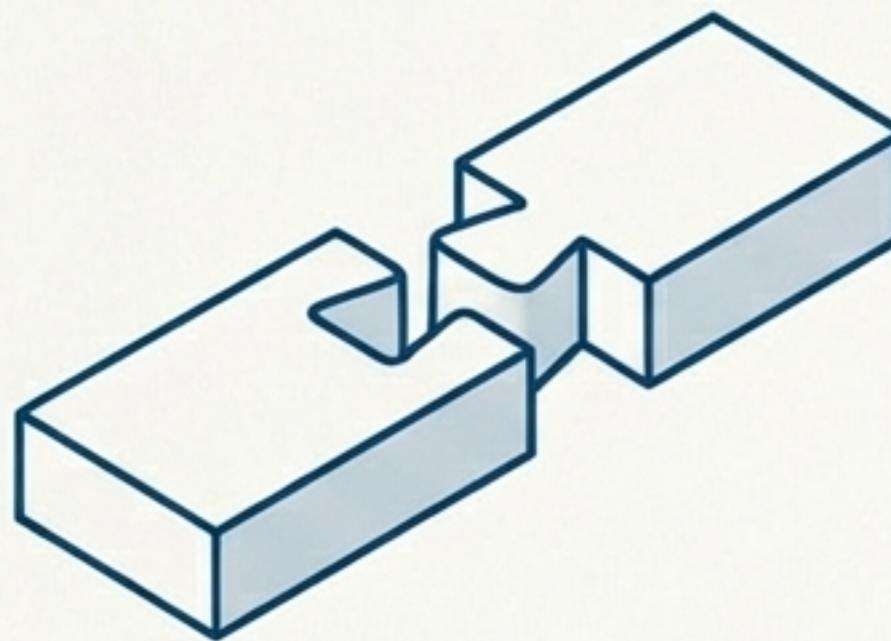
Desenhado para otimizar o desempenho no banco de dados SAP HANA.



## Foco em APIs

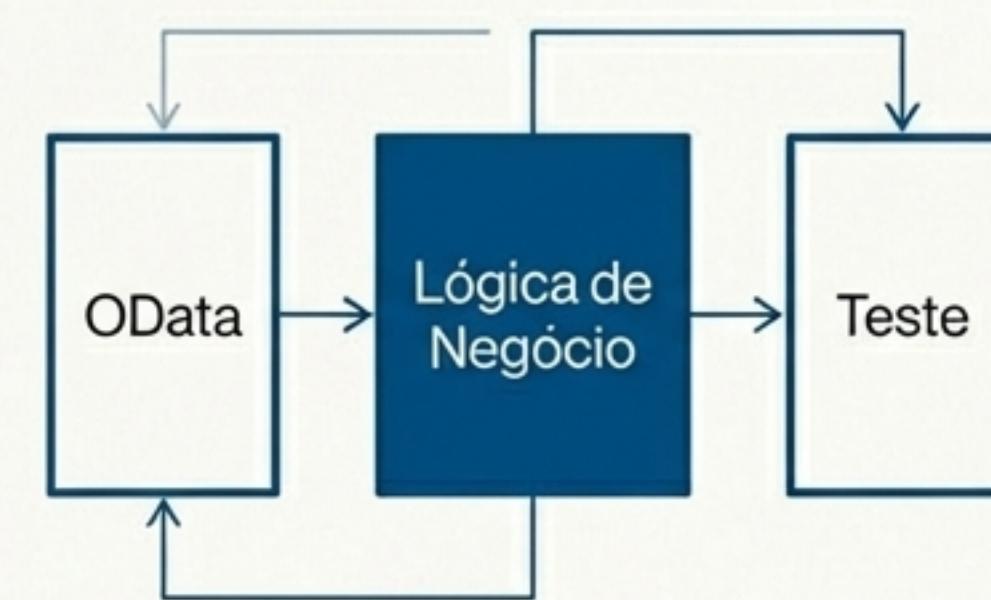
Permite a criação de serviços OData (APIs Web) robustos, otimizados e seguros para o ambiente corporativo.

# Os Pilares da Construção Moderna com RAP



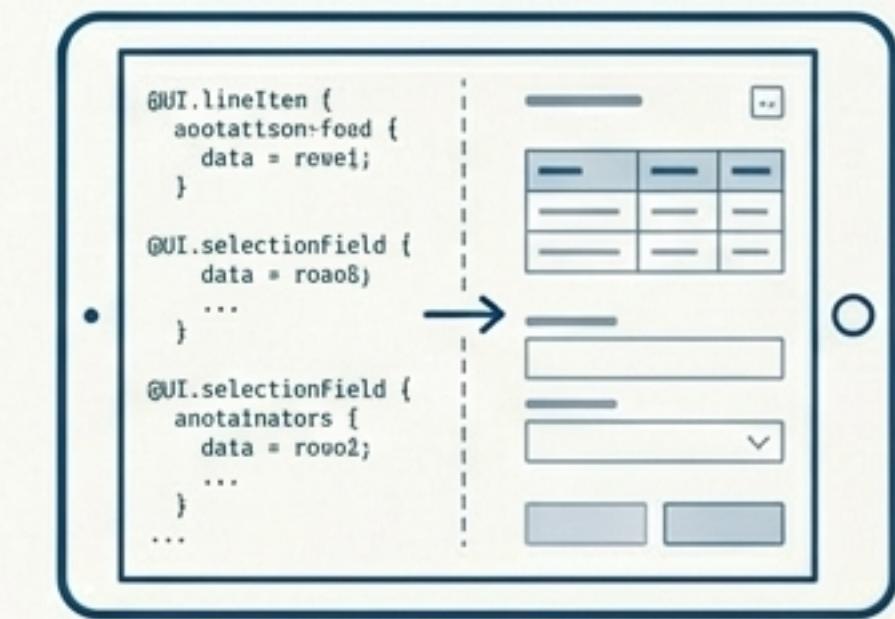
## Padronização

Define uma maneira única e clara de implementar operações transacionais (Create, Update, Delete), eliminando ambiguidades e aumentando a produtividade.



## Agnosticismo de Protocolo

A lógica de negócio é desacoplada do protocolo HTTP. Embora focado em OData, isso facilita testes unitários e a reutilização da lógica de negócios em diferentes contextos.



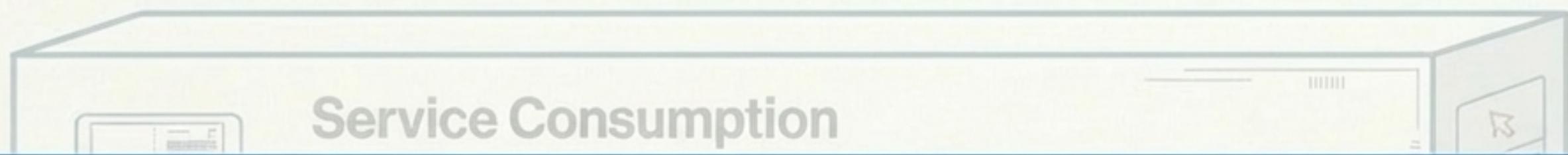
## Fiori Nativo

As aplicações RAP nascem com suporte nativo a anotações de UI. O SAP Fiori Elements lê esses metadados para gerar interfaces ricas e consistentes com mínimo esforço de front-end.

# A Planta Baixa do RAP: Uma Arquitetura em Três Camadas



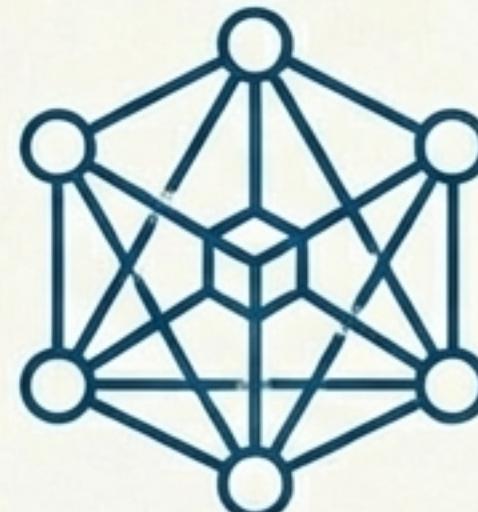
# Camada 1: O Coração – Modelagem de Dados e Comportamento



## CDS Views: O 'O Quê'

Definem o modelo de dados semântico (Virtual Data Model - VDM). Diferente de views SQL clássicas, as CDS Views contêm associações ricas, cálculos e metadados de domínio que descrevem a estrutura do negócio.

O 'Esqueleto' da aplicação.



## Behavior Definition (BDEF): O 'Como'

Artefato exclusivo do RAP que define o comportamento transacional. Declara explicitamente operações (Create, Update, Delete), validações, determinações (cálculos automáticos) e Ações customizadas (ex: 'Aprovar Pedido').

O 'Cérebro' da aplicação.



# Camada 2: A Exposição – Provisionamento de Serviços de Negócio



# Camada 3: O Consumo – A Aplicação em Uso



## SAP Fiori Elements

O framework lê os metadados e anotações do CDS para gerar UIs dinamicamente, sem necessidade de código JavaScript manual, resultando em apps consistentes e de alta qualidade.

## Web APIs

O serviço OData pode ser consumido por qualquer sistema externo (integrações A2A/B2B) ou por UIs customizadas (Freestyle SAPUI5, React, Angular), oferecendo máxima flexibilidade.

# A Grande Decisão do Arquiteto: Managed vs. Unmanaged

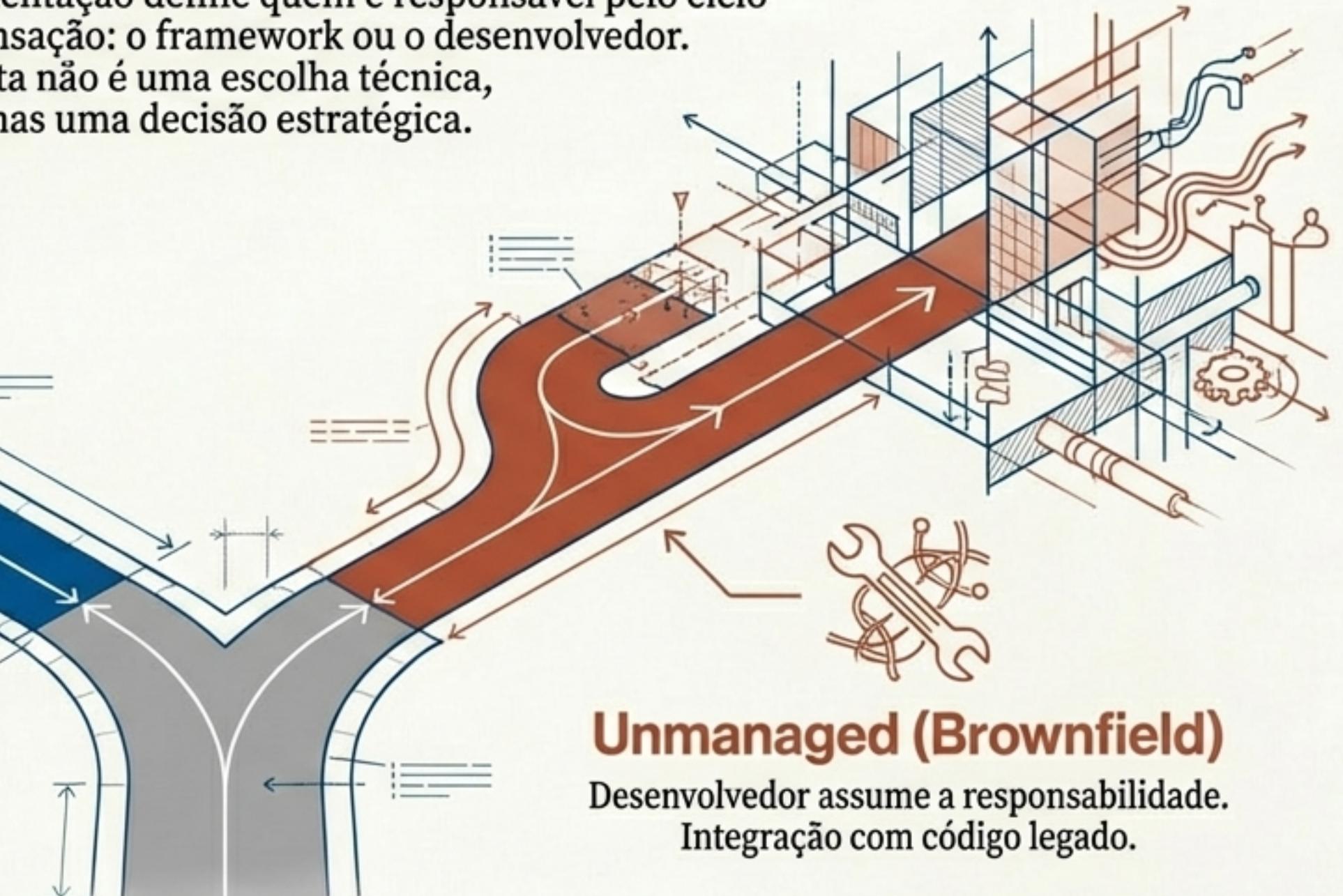
Ao criar um comportamento (Behavior Definition), a escolha do cenário de implementação define quem é responsável pelo ciclo de vida da transação: o framework ou o desenvolvedor.

Esta não é uma escolha técnica,  
mas uma decisão estratégica.



## Managed (Greenfield)

Framework assume a responsabilidade.  
Começando do zero.



## Unmanaged (Brownfield)

Desenvolvedor assume a responsabilidade.  
Integração com código legado.

Estamos construindo em um terreno novo ou integrando com uma estrutura existente?

# Piloto Automático vs. Modo Manual: Escolhendo seu Cenário



## Managed (Gerenciado)

O “Piloto Automático”

Blueprint  
Serif Pro

→ **Como funciona:** O framework RAP assume o controle total das operações CRUD padrão (INSERT, UPDATE, DELETE) e do bloqueio (locking).

**Papel do Desenvolvedor:** Focar exclusivamente na lógica de negócio específica: validações, determinações e ações.

Source  
Serif Pro

→ **Uso Ideal: Novos Desenvolvimentos (Greenfield).**  
Perfeito para tabelas novas, sem lógicas legadas complexas.



## Unmanaged (Não Gerenciado)

O “Modo Manual”

Source  
Serif Pro

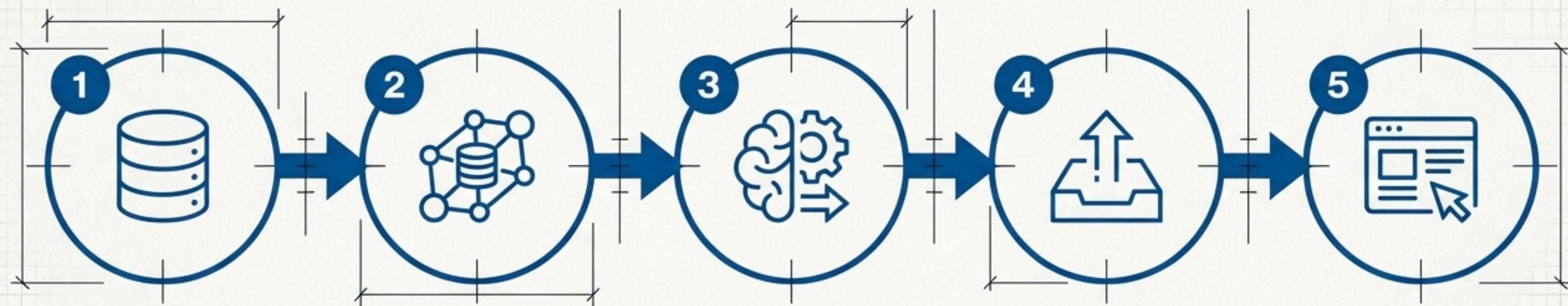
→ **Como funciona:** O framework delega a responsabilidade da persistência. O desenvolvedor implementa os métodos para salvar os dados.

**Papel do Desenvolvedor:** Escrever o código que chama BAPIs, Funções ou classes legadas. Gerenciar o buffer transacional.

Source  
Serif Pro

→ **Uso Ideal: Sistemas Legados (Brownfield).** Essencial para construir um app moderno sobre uma lógica de negócio existente e complexa (ex: BAPI de criação de ordem de venda).

# O Canteiro de Obras: O Fluxo de Desenvolvimento do Zero ao Pro Desenvolvimento do Zero ao App



## Base de Dados

Criação da Tabela Física

## Modelo de Dados

Construção das CDS Views  
(Interface e Projeção)

## Comportamento

Definição da Lógica Transacional  
(Behavior Definition)

## Exposição

Criação do Serviço  
(Service Definition & Binding)

## Consumo

Pré-visualização e Teste da UI Fiori

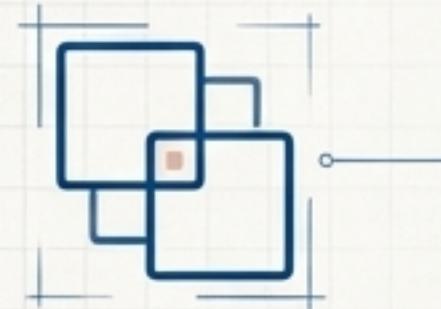
# Fluxo de Trabalho em Detalhes (Parte 1): A Fundação



## 1. Database Table

Criação da tabela física no Dicionário ABAP.

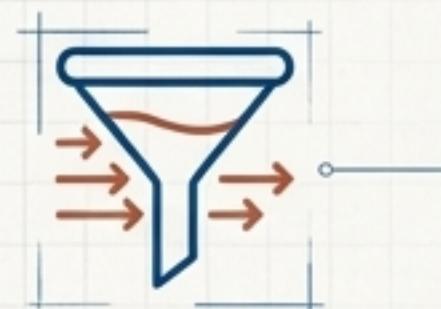
**Boa Prática:** Utilizar UUIDs como chaves primárias e campos de controle administrativo.



## 2. Interface View (CDS)

Criação da visão básica (Basic View) sobre a tabela.

**Propósito:** Abstrair os nomes técnicos da tabela para nomes semânticos (CamelCase).



## 3. Projection View (CDS)

Criação de uma visão de consumo (Consumption View) para a aplicação.

**Propósito:** Filtrar campos, preparar dados e servir como base para a UI.

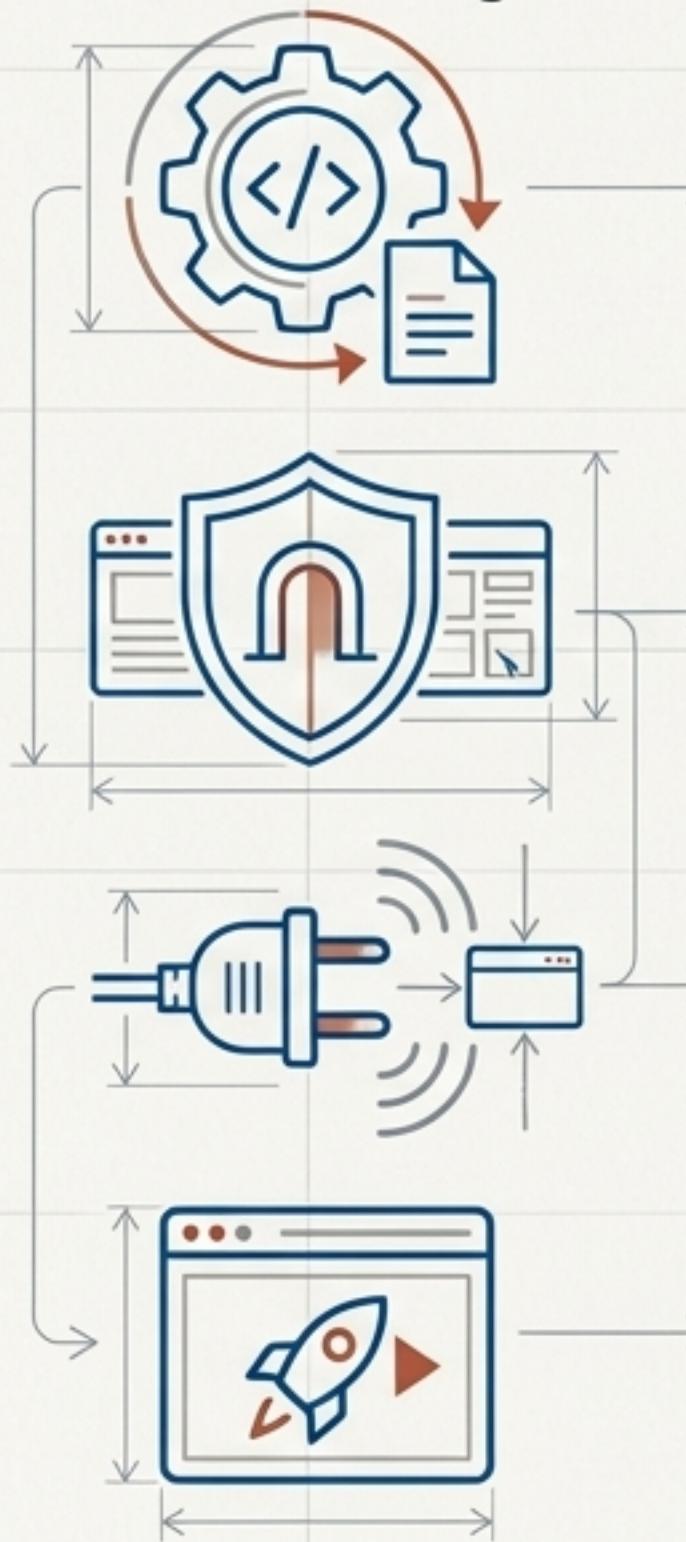


## 4. Metadata Extension

Criação de um arquivo separado (.ddlx) para as anotações de UI (@UI.lineItem, etc.).

**Benefício:** Mantém a CDS View limpa, separando a lógica de dados da lógica de apresentação.

# Fluxo de Trabalho em Detalhes (Parte 2): A Publicação



## 5. Behavior Definition (BDEF)

Definir as capacidades transacionais na Interface View e projetá-las na Projection View.

**Resultado:** O compilador cria as classes (Behavior Pools) onde o código ABAP será implementado.

## 6. Service Definition

Agrupar as Projection Views relevantes em um serviço nomeado e com escopo definido.

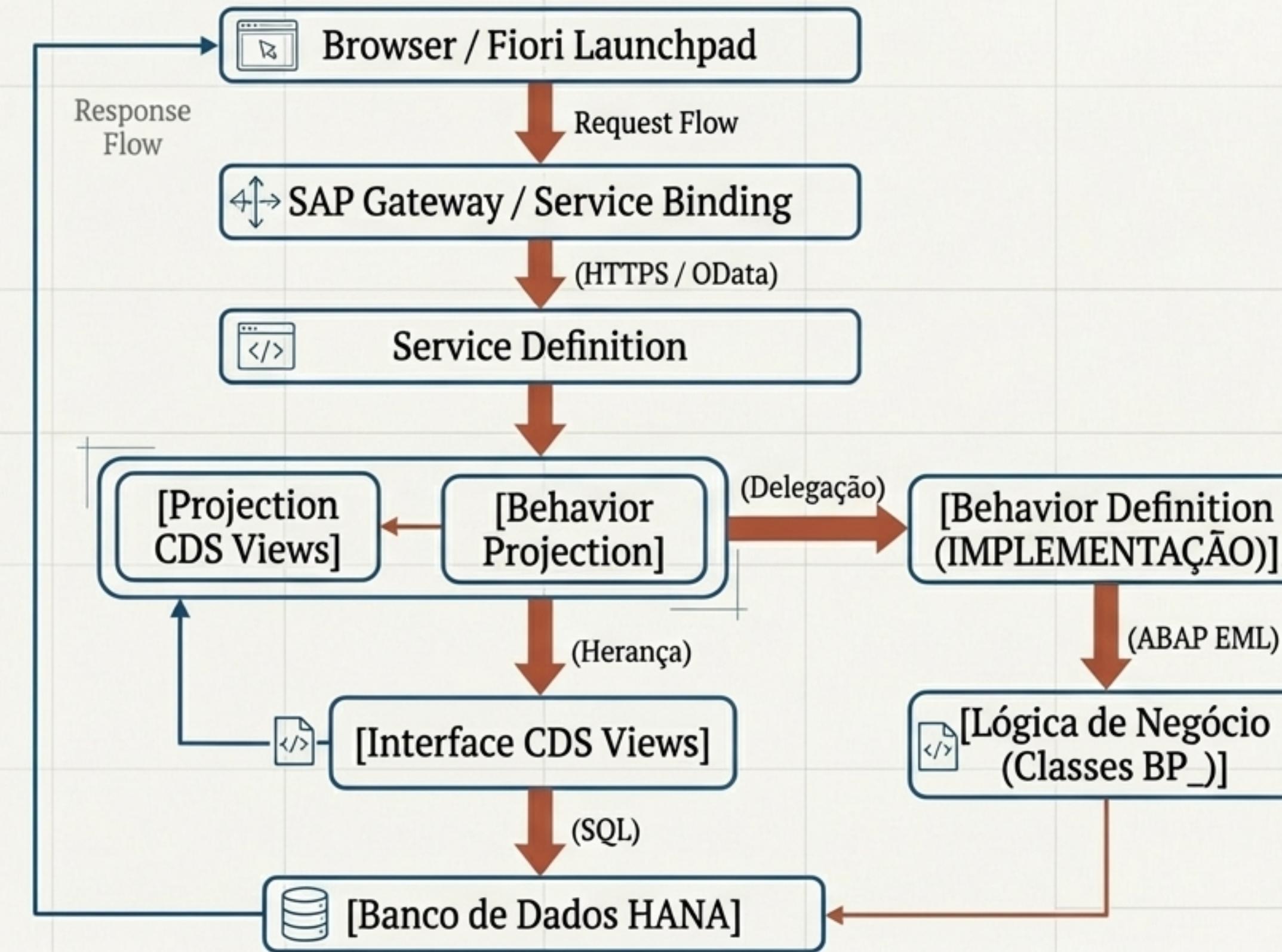
## 7. Service Binding

Publicar o serviço, escolhendo o protocolo e ativando o endpoint (/sap/opu/odata/...).

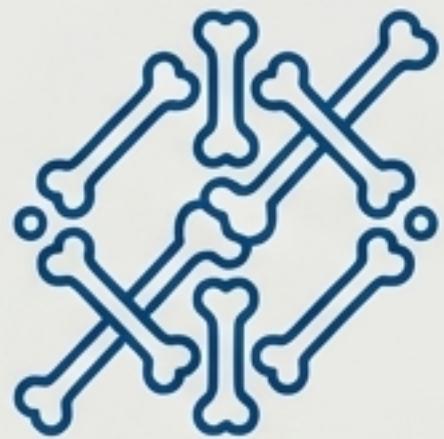
## 8. Preview

**Destaque:** Utilizar a função de *Preview* do Service Binding no ADT (Eclipse) para testar a aplicação Fiori Elements instantaneamente no navegador.

# O Fluxo de Dados Consolidado



# Resumo do Arquiteto: Conceitos Essenciais em um Relance



## CDS View

Função: Modelagem de Dados

Camada: Data Modeling

Analogia (in Terracotta (#B45339)): O "Esqueleto" e os "Músculos"



## Behavior Definition

Função: Lógica Transacional

Camada: Business Logic

Analogia (in Terracotta): O "Cérebro" (Decisões e Regras)



## Service Definition

Função: Seleção de Escopo

Camada: Service Provisioning

Analogia (in Terracotta): O "Cardápio" (O que está disponível)



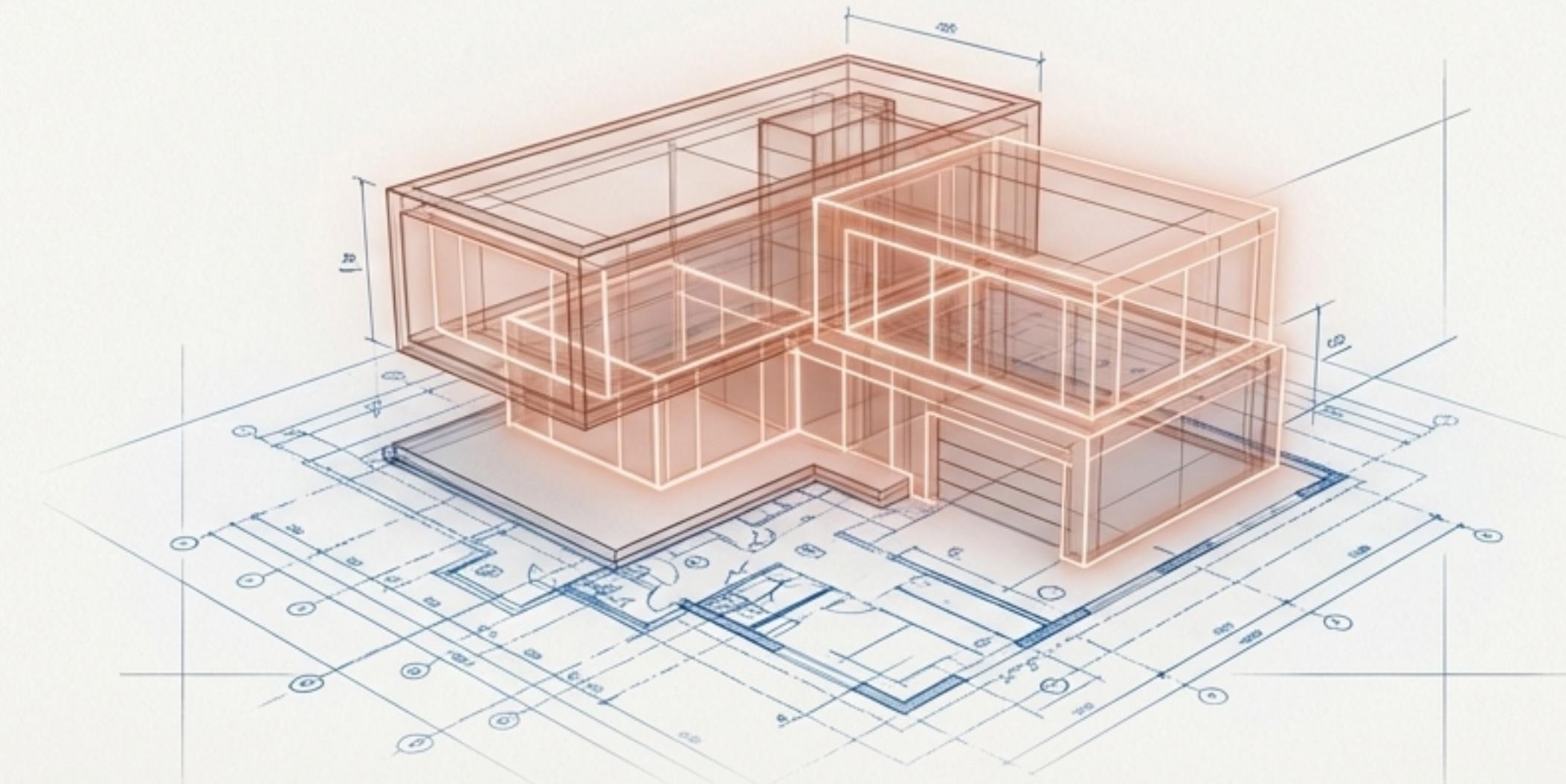
## Service Binding

Função: Protocolo de Comunicação

Camada: Service Provisioning

Analogia (in Terracotta): O "Garçom" (Entrega o pedido em OData)

# Construindo o Futuro com ABAP RAP



O ABAP RESTful Application Programming Model é a base para o desenvolvimento de aplicações S/4HANA e ABAP Cloud. Dominar sua arquitetura, decisões e fluxo de trabalho é essencial para qualquer arquiteto de software SAP moderno.

**A planta baixa foi apresentada. As ferramentas estão prontas. Você está pronto para construir.**