



Clean Architecture

A Clean Architecture é um modelo que tem como objetivo o desacoplamento entre as regras de negócio, ou domínio, da aplicação e os recursos externos como frameworks e banco de dados

FrontPage

Not Secure | mail.fitnesse.org/FrontPage

 FitNesse Features Download Plug-ins User Guide

FrontPage

The fully integrated standalone wiki and acceptance testing framework

Be-monster not thy feature, wer't my fitness
– Shakespeare, King Lear

[Download FitNesse](#)

It's a Collaboration tool

Since FitNesse is a [wiki web server](#), it has a very low entry and learning curve, which makes it an excellent tool to collaborate with, for example, business stakeholders.

[Read more...](#)

It's a Test tool

The wiki pages created in FitNesse are run as tests. The specifications can be tested against the application itself, resulting in a roundtrip between specifications and implementation.

[Read more...](#)

It's Open

FitNesse is an open source project. The code base is not owned by any company. A lot of information is shared by the FitNesse community. It's extremely adaptable and is used in areas ranging from Web/GUI tests to testing electronic components.

[Read more...](#)

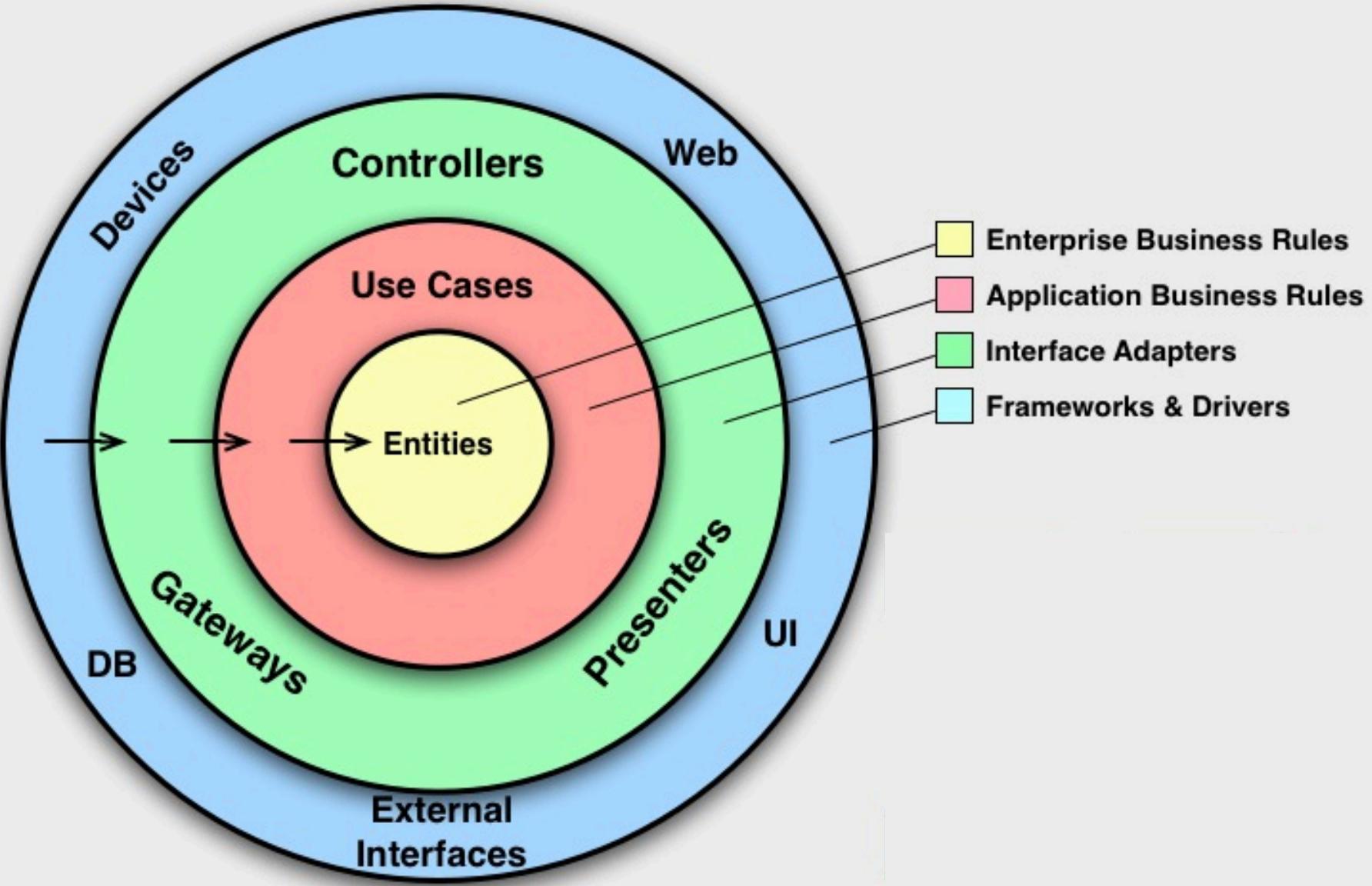
FitNesse v20201213

[From Page](#) - [User Guide](#) - [Features](#) - [Download](#) - [Plugins](#) - [Fixtures](#) - [Stark Community](#)

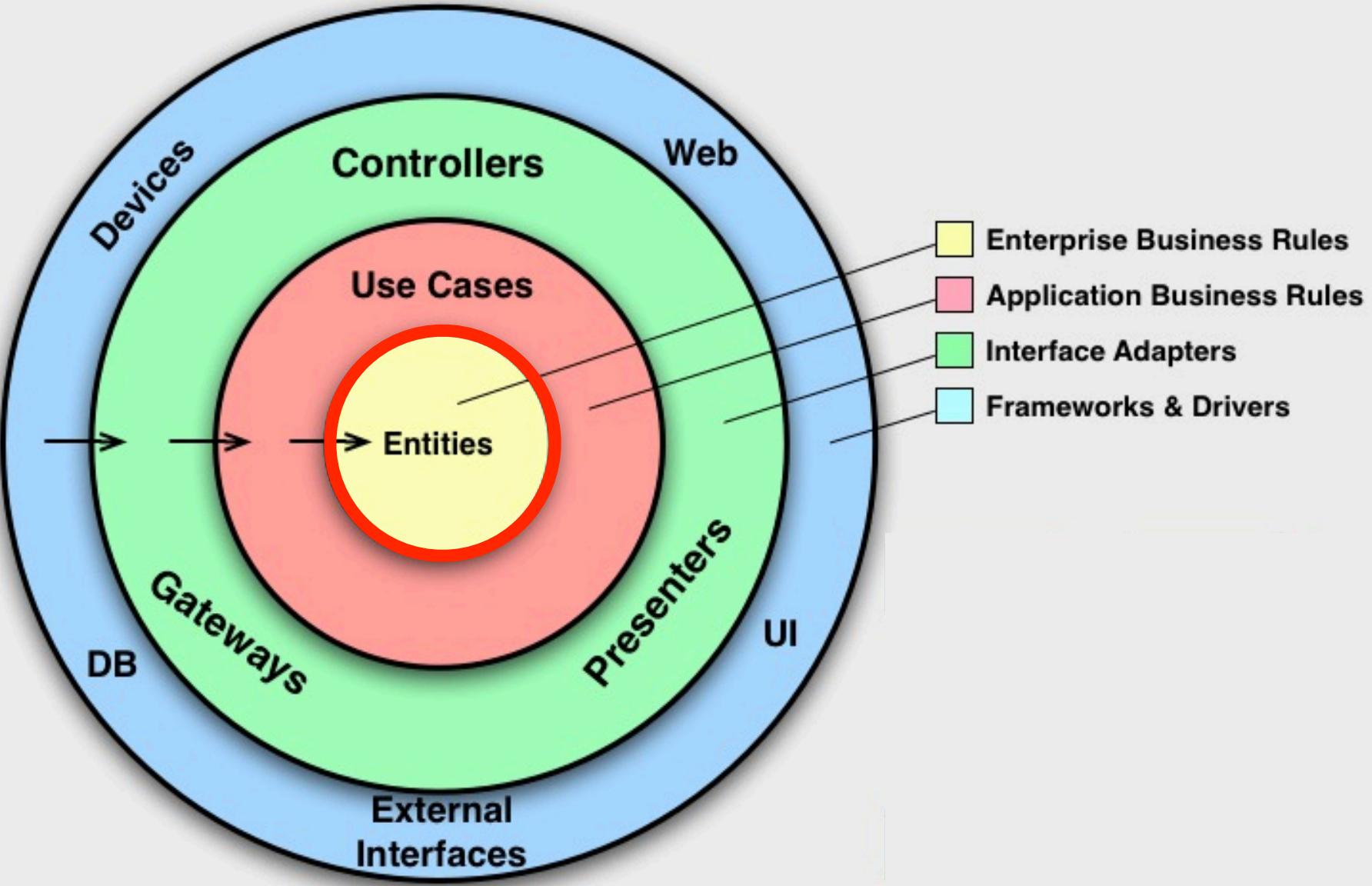
"The center of your application is not the database, nor is it one or more of the frameworks you may be using. The center of your application is the use cases of your application"

Robert Martin

The Clean Architecture



The Clean Architecture



Entidades são responsáveis por abstrair as **regras de negócio independentes**, que podem ser desde um objeto com métodos até mesmo um conjunto de funções

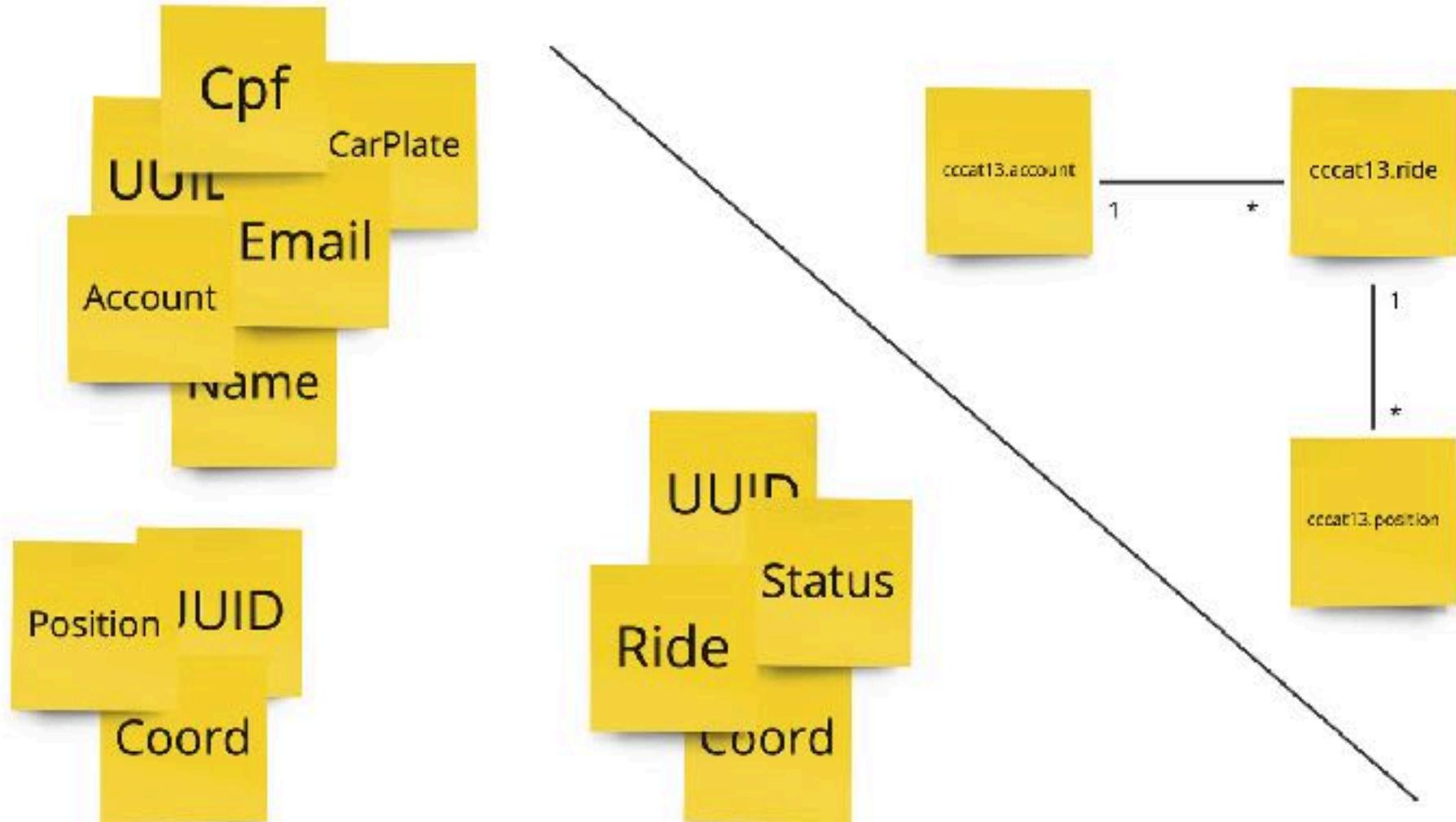
O que são regras de negócio independentes?

- O cpf é válido?
- O nome é válido?
- O email é válido?
- A placa do carro é válida?
- Qual é a distância entre duas coordenadas?
- Quanto é a tarifa da corrida?
- Como é a mudança de status de uma corrida?



CAUTION

Essas entidades não são as mesmas que
utilizamos em um **ORM**





O problema do domínio anêmico

Transaction Script

Organizes business logic by procedures where each procedure handles a single request from the presentation.

For a full description see P of EAA page 110

```
recognizedRevenue(contractNumber: long, asOf: Date) : Money  
calculateRevenueRecognitions(contractNumber long) : void
```

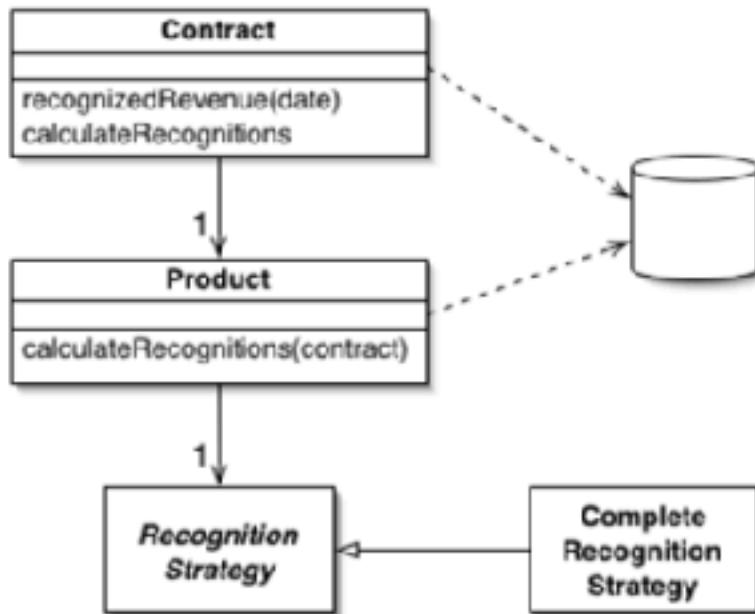
Most business applications can be thought of as a series of transactions. A transaction may view some information as organized in a particular way, another will make changes to it. Each interaction between a client system and a server system contains a certain amount of logic. In some cases this can be as simple as displaying information in the database. In others it may involve many steps of validations and calculations.

A Transaction Script organizes all this logic primarily as a single procedure, making calls directly to the database or through a thin database wrapper. Each transaction will have its own Transaction Script, although common subtasks can be broken into subprocedures.

Domain Model

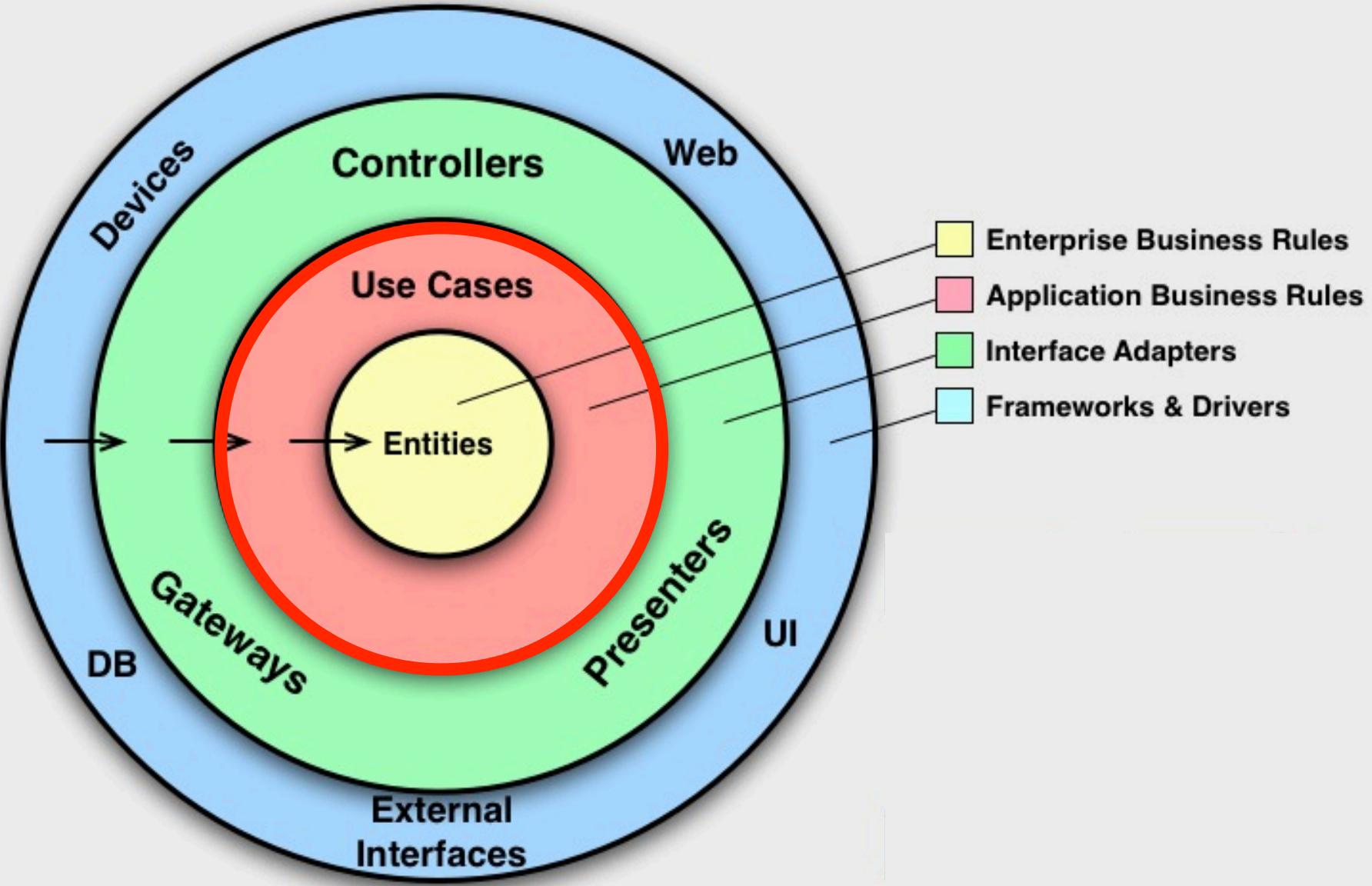
An object model of the domain that incorporates both behavior and data.

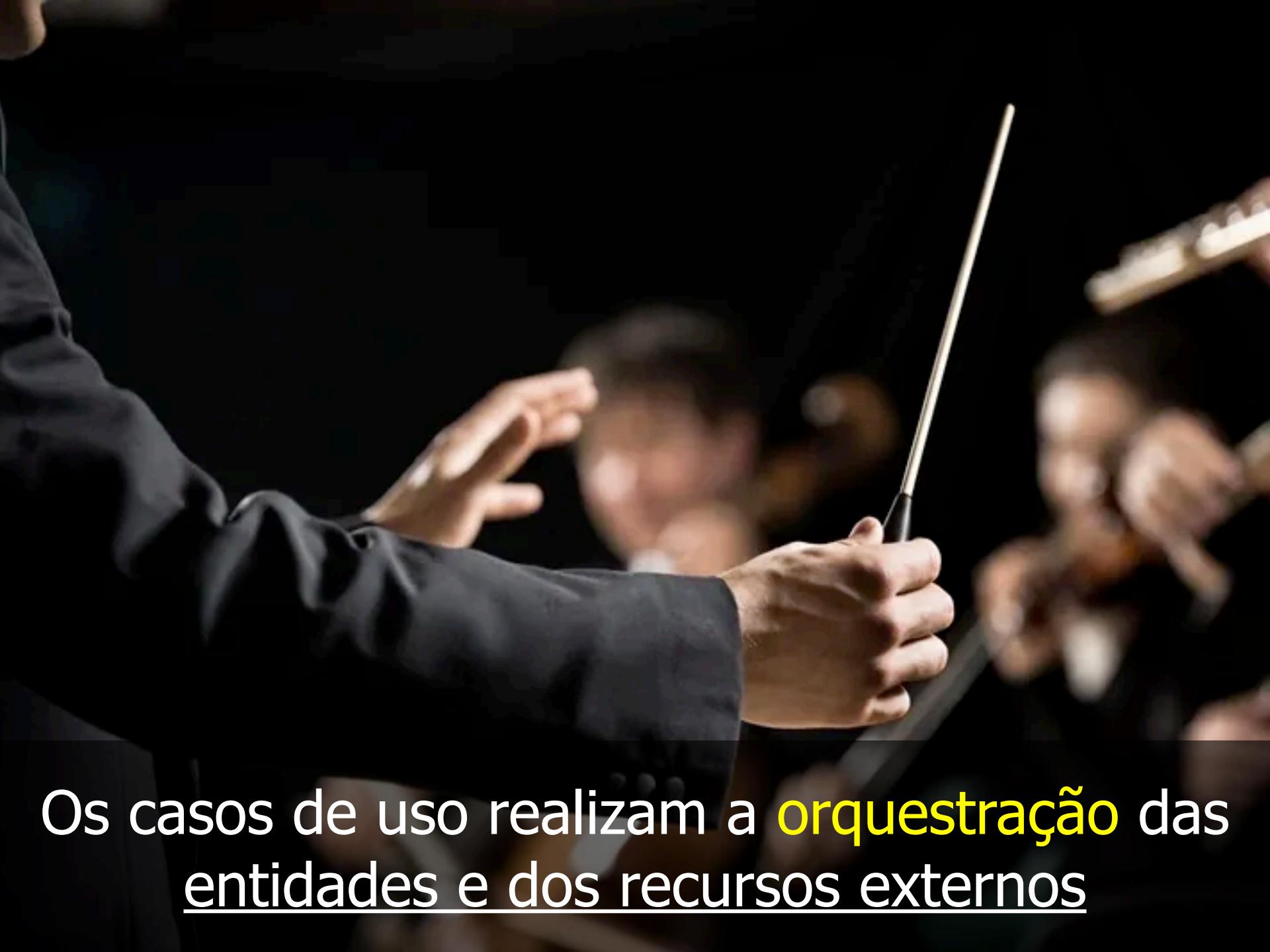
For a full description see P of EAA page 116



At its worst business logic can be very complex. Rules and logic describe many different cases and slants of behavior, and it's this complexity that objects were designed to work with. A Domain Model creates a web of interconnected objects, where each object represents some meaningful individual, whether as large as a corporation or as small as a single line on an order form.

The Clean Architecture



A close-up photograph of a conductor's hands and arms. The conductor is wearing a dark suit jacket. One hand holds a baton, which is positioned vertically. The other hand is raised, fingers spread, in a gesture of指挥 (conducting). In the blurred background, musicians playing string instruments like violins and cellos are visible, suggesting a live performance.

Os casos de uso realizam a **orquestração** das entidades e dos recursos externos

Use Cases

- Criar uma conta
- Validar uma conta
- Solicitar uma corrida
- Aceitar uma corrida
- Iniciar uma corrida
- Atualizar a posição da corrida
- Finalizar a corrida

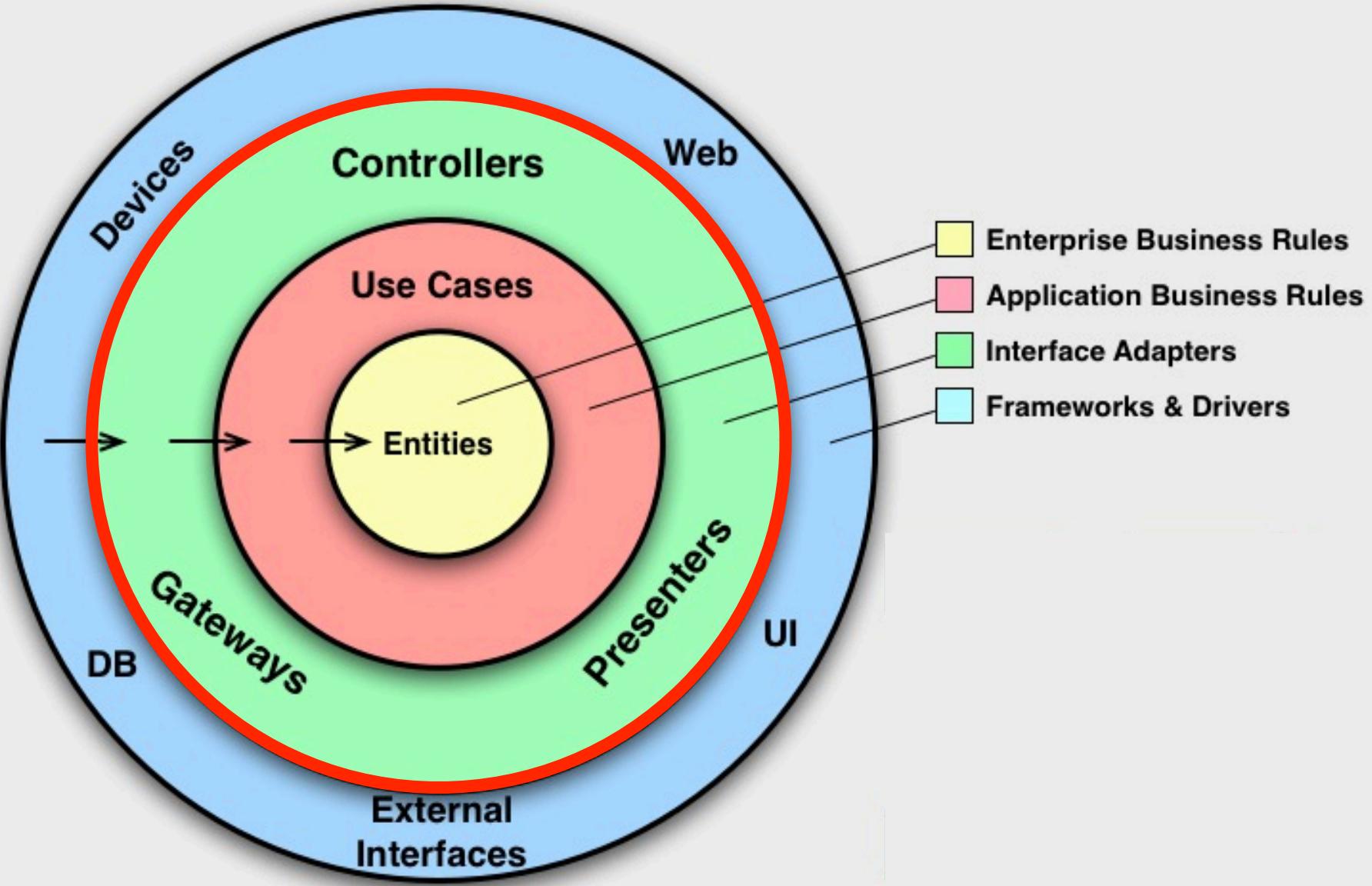


Reparam que os nomes dos casos de uso tem
relação com a **Screaming Architecture**



Caso de uso é **diferente** de CRUD?

The Clean Architecture





Os interface adapters fazem a **ponte** entre os casos de uso e os recursos externos

Tratamento de requisições e respostas
HTTP, lidando com parâmetros

ARE

CURSOR cursorvalue

```
SELECT h.procedure_name
FROM company o
WHERE o.procedure_name = 'P'
ORDER BY 2
```

Acesso ao banco de dados, todo o código SQL pertence à esta camada



Integração com uma **API** externa

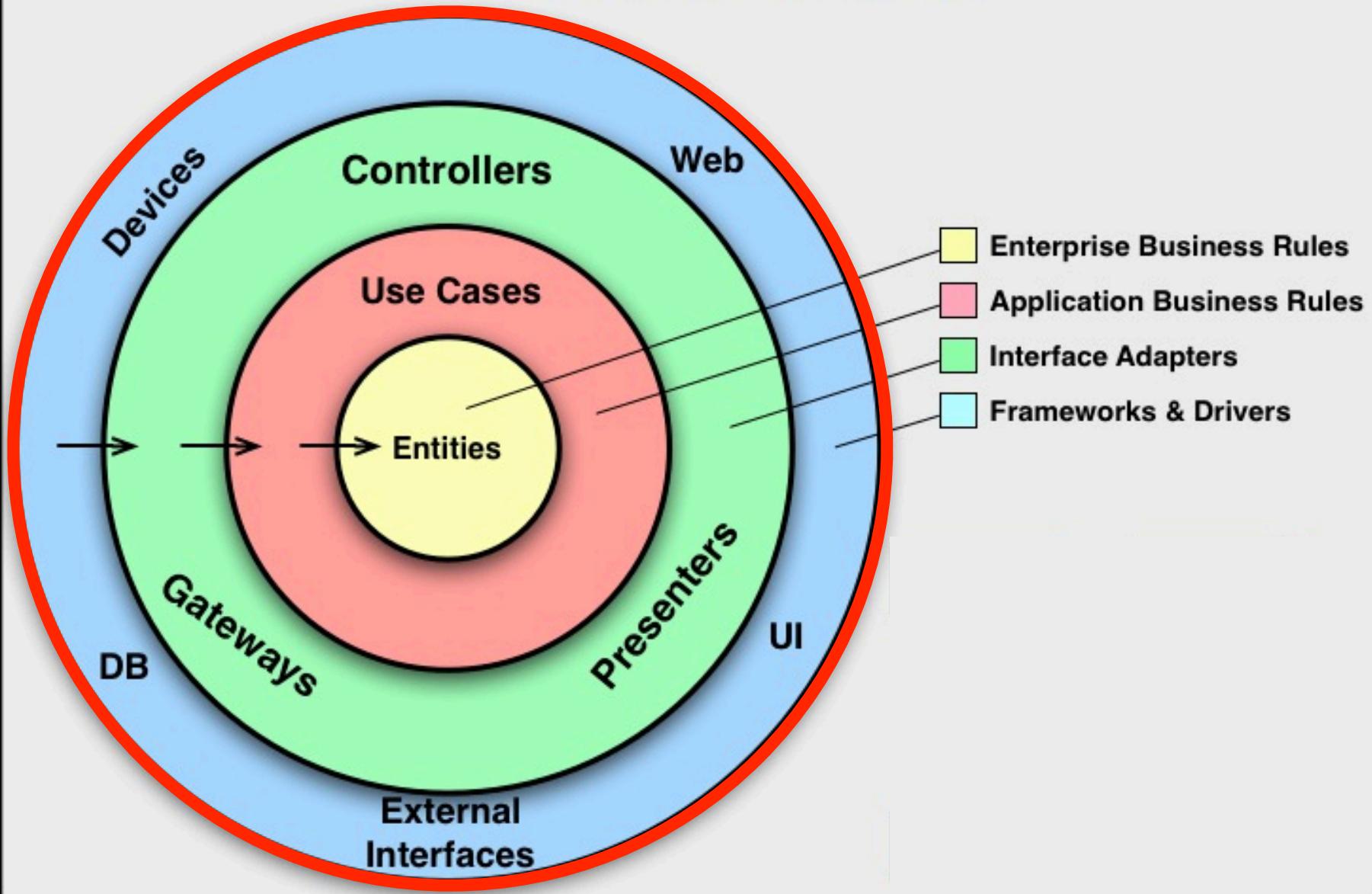


Escrita e leitura no sistema de arquivos



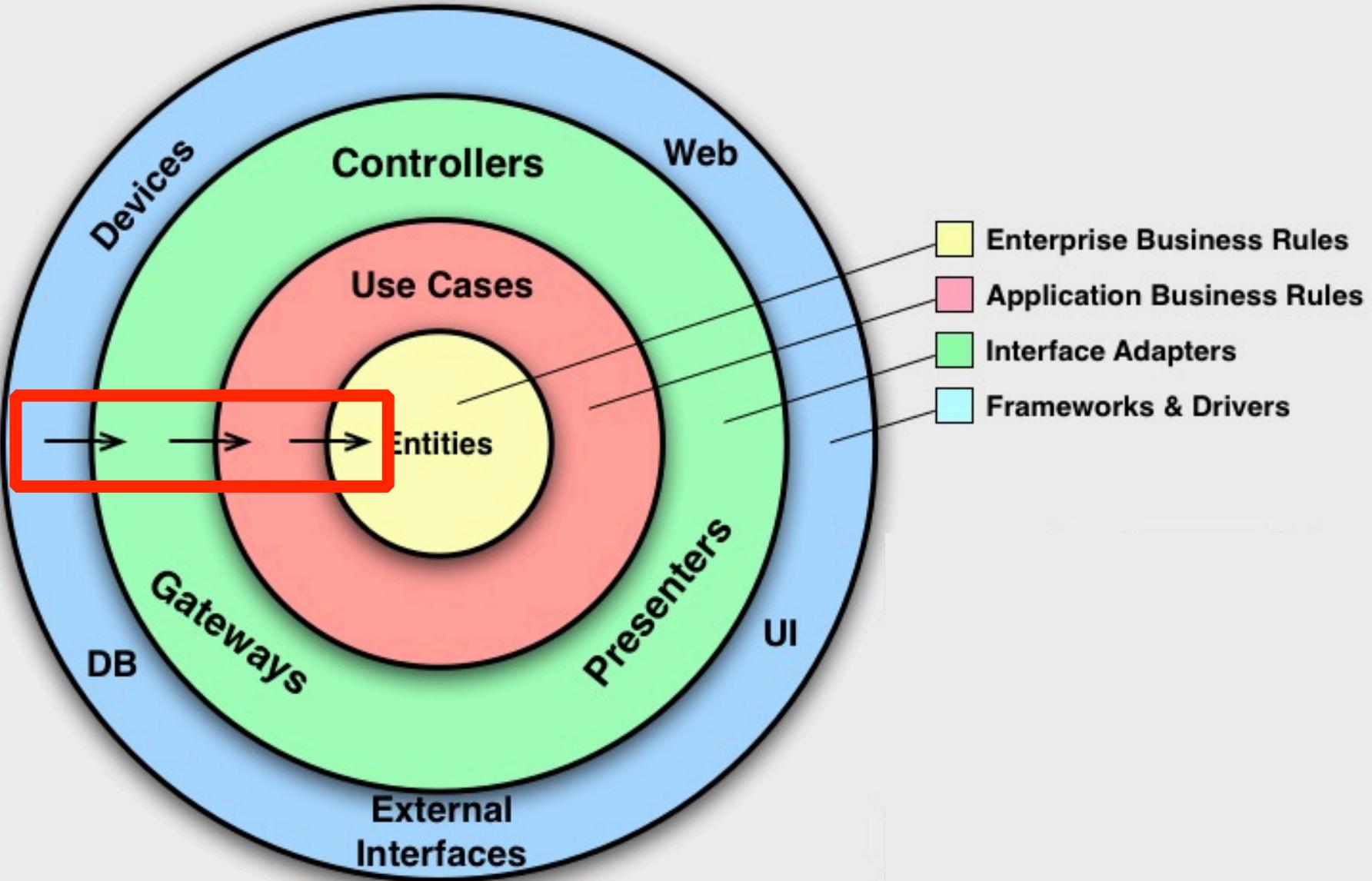
**Conversão de dados para formatos
específicos como CSV e PDF**

The Clean Architecture

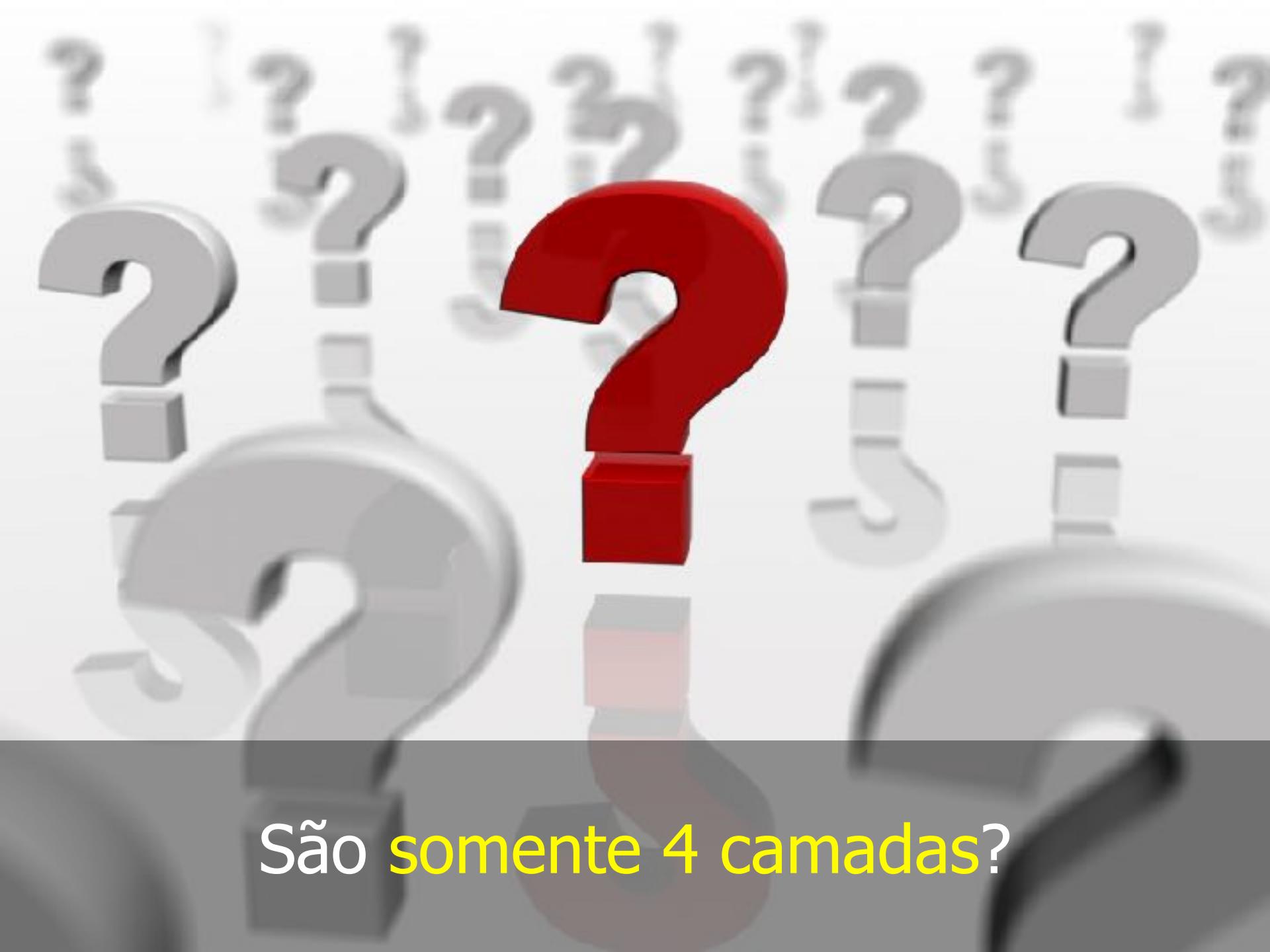


Por fim, os frameworks and drivers são o nível mais baixo de abstração, é a interação com a tecnologia, com os componentes que realizam a conexão com o banco de dados, as requisições HTTP, a interação com o sistema de arquivos ou o acesso aos recursos do sistema operacional

The Clean Architecture



**Quem está dentro não conhece quem está fora,
mas quem está fora conhece quem está dentro,
as entidades não conhecem os use cases e esses
não conhecem a implementação dos interface
adapters, que não conhecem a implementação
dos frameworks and drivers**



São somente 4 camadas?

Uma camada é uma **fronteira lógica** entre um conjunto de componentes, que tem uma responsabilidade bem definida



As camadas não tem necessariamente
relação física com as pastas



Como incializamos a aplicação?

O **main** é o ponto de entrada da aplicação (HTTP, CLI, UI, Testes), é lá que as fábricas e estratégias são inicializadas e as injeções de dependência são realizadas durante a inicialização

When composing an application from many loosely coupled classes, **the composition should take place as close to the application's entry point as possible**. The Main method is the entry point for most application types. The Composition Root composes the object graph, which subsequently performs the actual work of the application

