

# TÓPICO 13 - ARQUITETURA LIMPA

Clean Code - Professor Ramon Venson - SATC 2025.2

## O que é arquitetura?

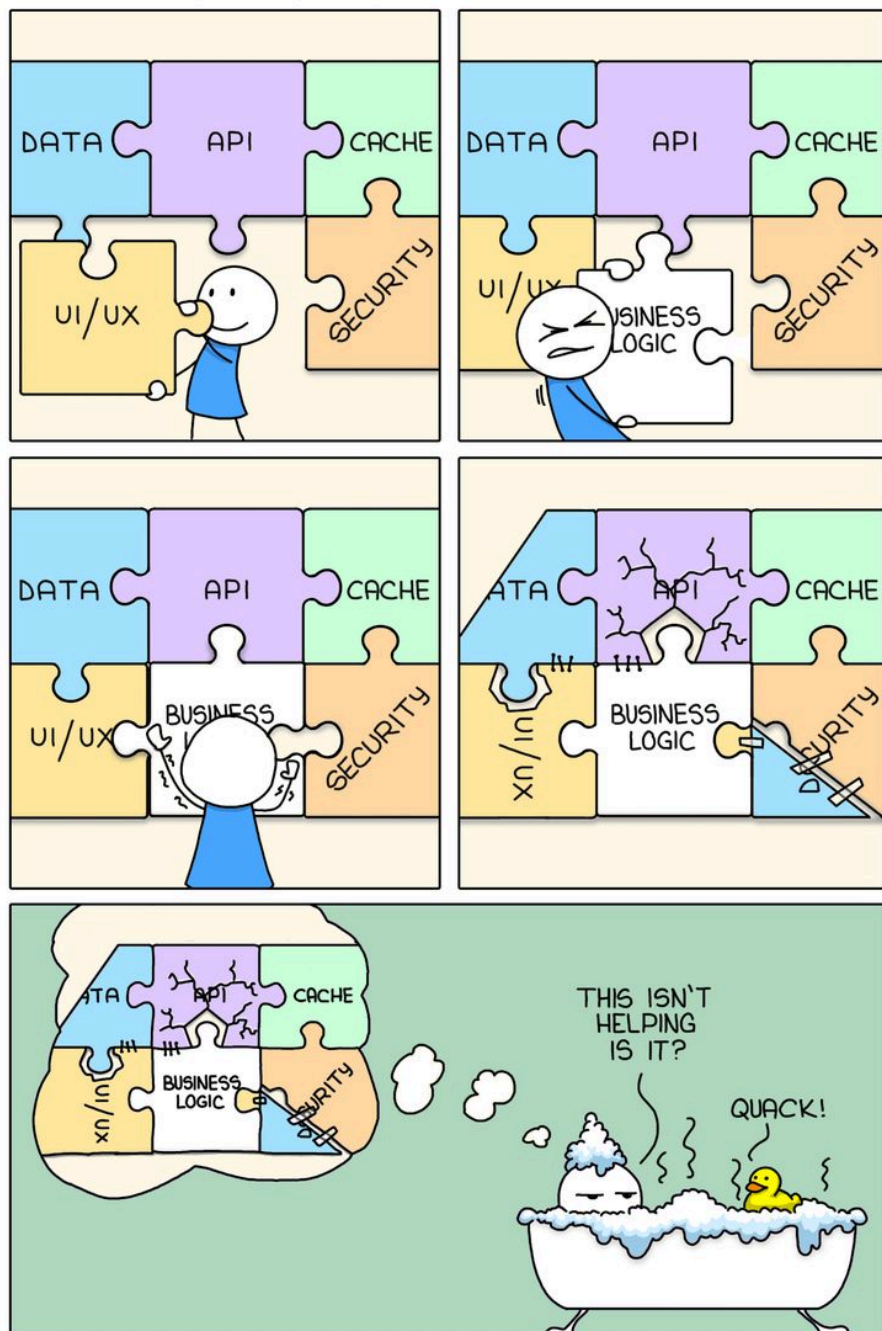
- Hardware : como x86, ARM, RISC-V;
- Processador : como Intel, AMD, Apple M1;
- Sistema Operacional : como tempo compartilhado, real-time, multitarefas;
- Rede : Peer-to-Peer, Cliente-Servidor;
- Nuvem : IaaS, PaaS, SaaS, FaaS;
- Arquitetura de Software : como MVC, Clean Architecture, Hexagonal, Monolítica, Microserviços, etc...

## O que é arquitetura de software?

Arquitetura de software é uma estrutura intencional para gerenciar complexidade e mudança em software.

Não importa o tamanho do projeto, uma arquitetura de software tem como objetivo responder as seguintes perguntas:

- Do que o sistema é composto?
- Como esses componentes se relacionam?
- Onde estão as decisões?
- Podemos mudar algo sem afetar outras partes do sistema?



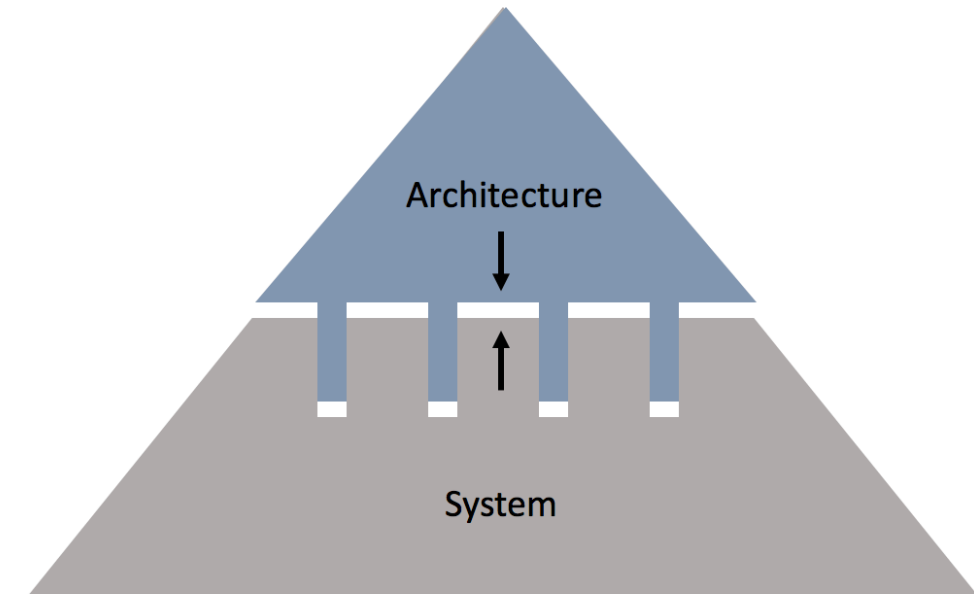
### Arquitetura do software garante:

- **Entendimento** : o que o sistema faz e como ele funciona.
- **Evolução** : como podemos mudar o sistema sem afetar outras partes?
- **Operação** : o que o sistema precisa para ser operado?

## Aspectos de Arquitetura

Uma arquitetura pode ser dividida entre dois aspectos conceituais:

- 📦 **Arquitetura de Aplicação:** como o sistema é estruturado?
- ⚙️ **Arquitetura de Sistema:** como o sistema interage em ambiente de execução?



## Arquitetura Lógica

Como o sistema é estruturado?

A arquitetura lógica é a parte mais concreta da arquitetura.

Ela define como vamos organizar o sistema em classes, pacotes, módulos, etc.

Arquiteturas como Domain-Driven Design , Screaming Architecture e Event Sourcing , Hexagonal Architecture , Onion Architecture , Clean Architecture , Model-View-Controller são exemplos de decisões que definem a estrutura do sistema.

## Domain-Driven Design

Modelar o software de acordo com o domínio do problema real, usando linguagem comum entre técnicos e especialistas do negócio.

## Screaming Architecture

Pastas e módulos  
refletem a intenção  
comercial (por exemplo,  
/faturamento/ ao invés de  
/serviços )

## Event Sourcing

A arquitetura de eventos é uma abordagem que permite que o estado do sistema seja reconstruído a partir de uma sequência de eventos.

## Hexagonal Architecture

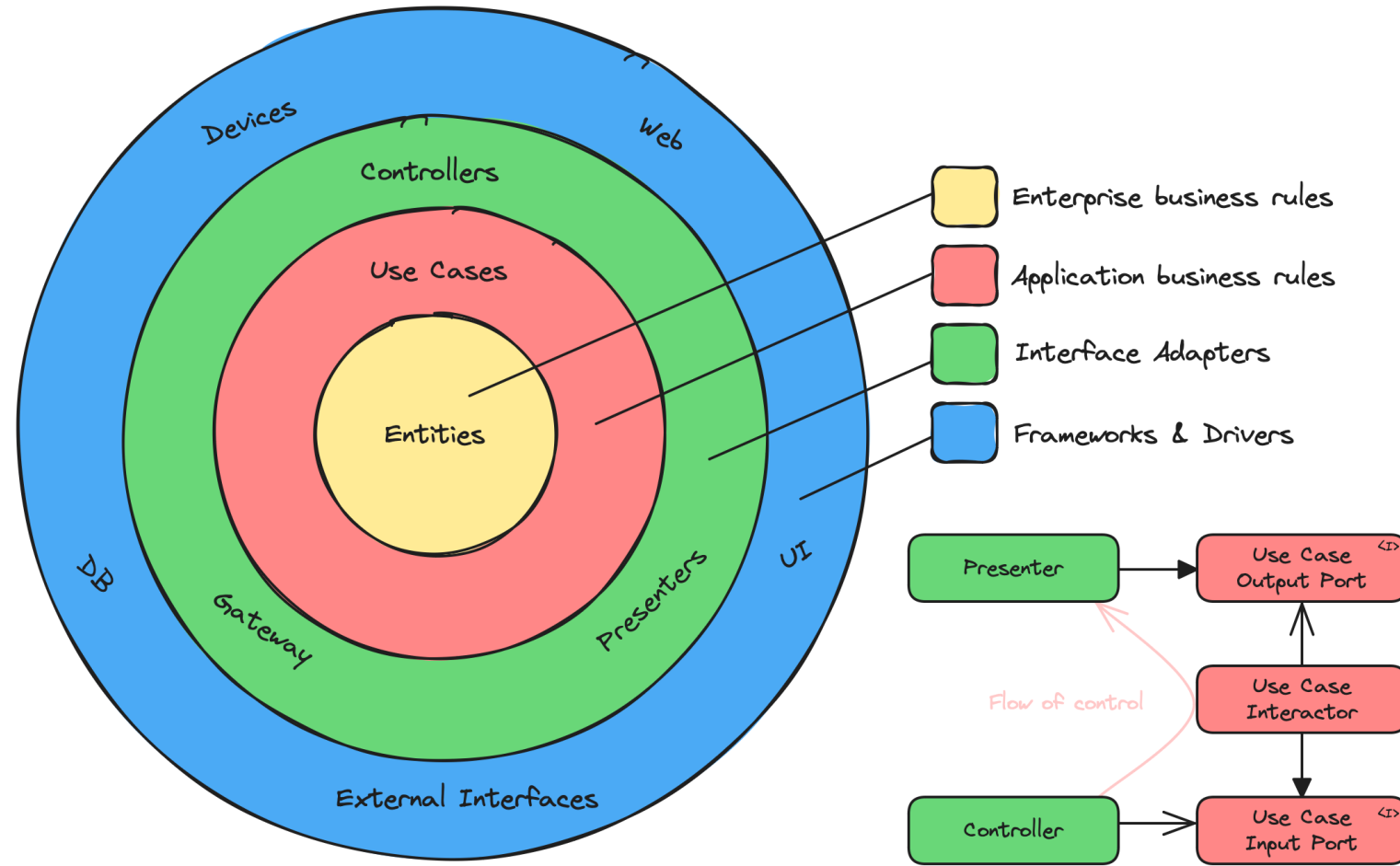
Desacoplar o núcleo da aplicação dos elementos externos (UI, BD, APIs) usando portas (interfaces) e adaptadores (implementações).

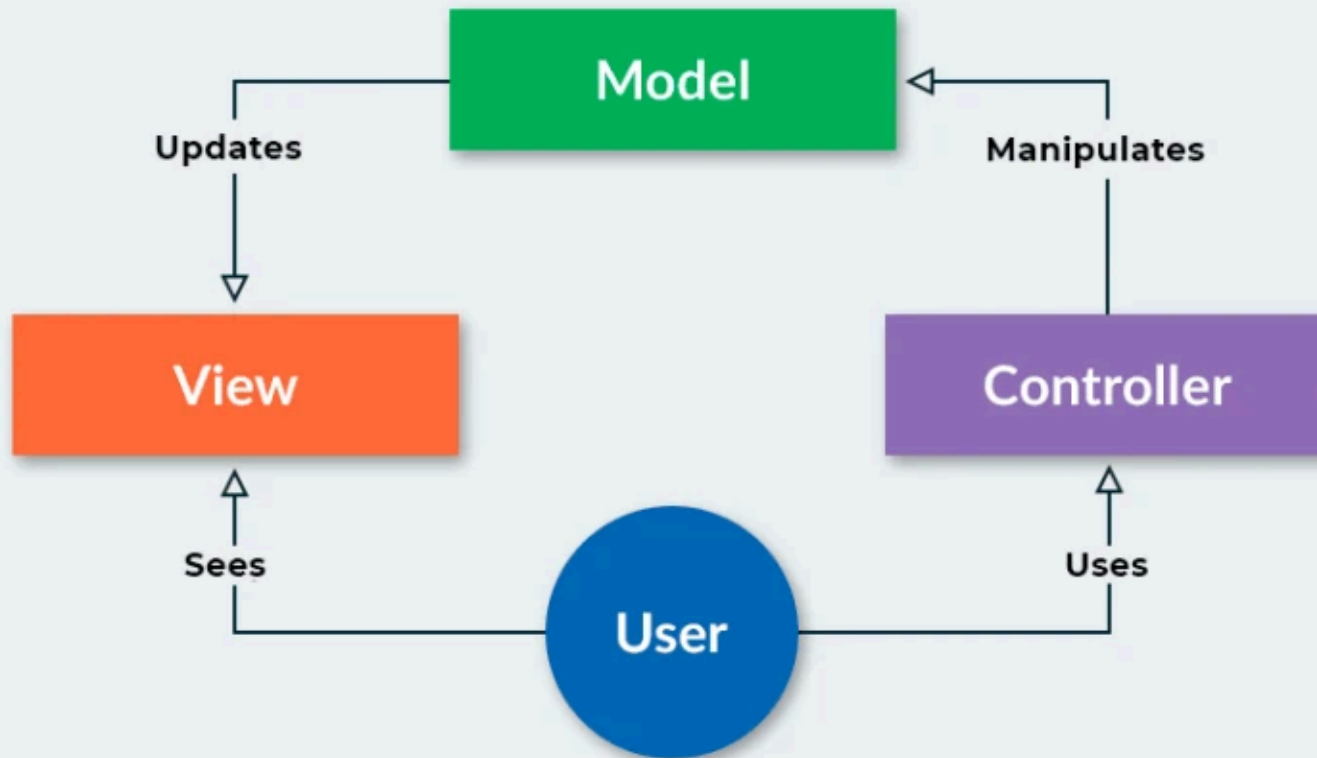
## Onion Architecture

Modelar o sistema em anéis concêntricos (como uma cebola) com o modelo de domínio no centro, enfatizando a separação de responsabilidades.

## Clean Architecture

Organiza o software em camadas concêntricas com clara separação entre regras de negócio e detalhes técnicos. As camadas internas representam a lógica central e são independentes de frameworks externos.





## Model-View-Controller

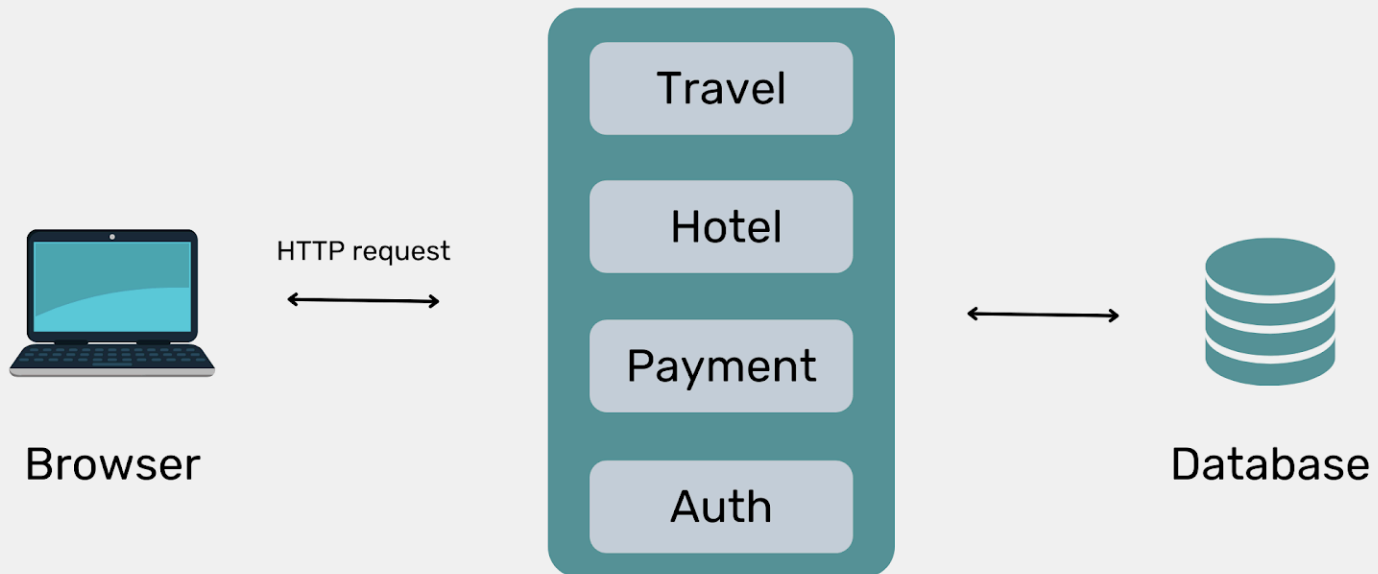
Separar a lógica de apresentação da lógica de negócios e da manipulação de entrada do usuário.

## Arquitetura Física

A parte física da arquitetura define como o sistema será executado em ambiente de execução.

Arquiteturas como `Microservices` , `Monolitos` , `Serverless` são exemplos de decisões que definem a forma de implantação do sistema.

## Monolithic Architecture



## Monolitos

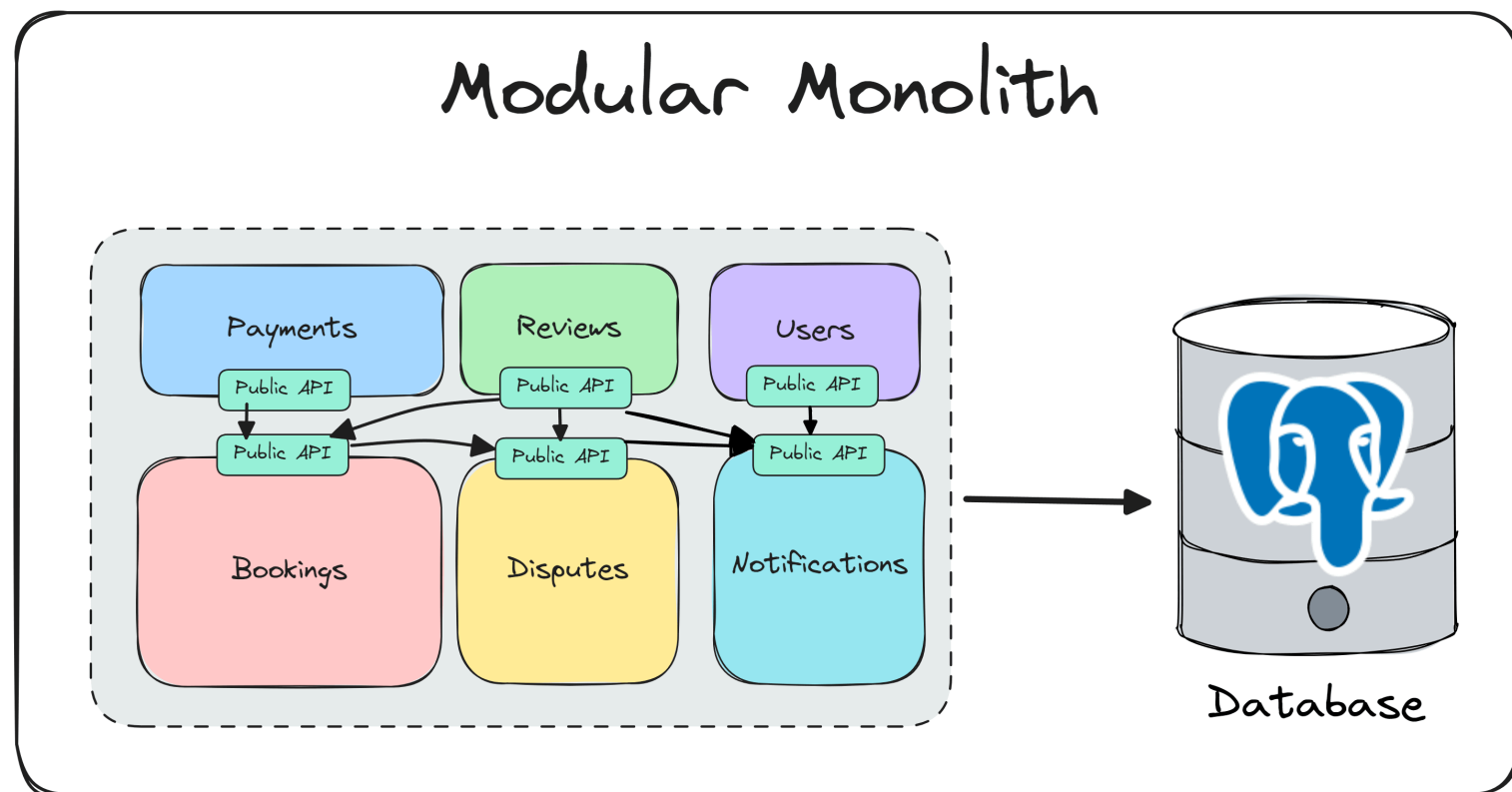
Monolito é uma arquitetura de software que consiste em um único serviço que contém toda a lógica de negócio.

## Microservices

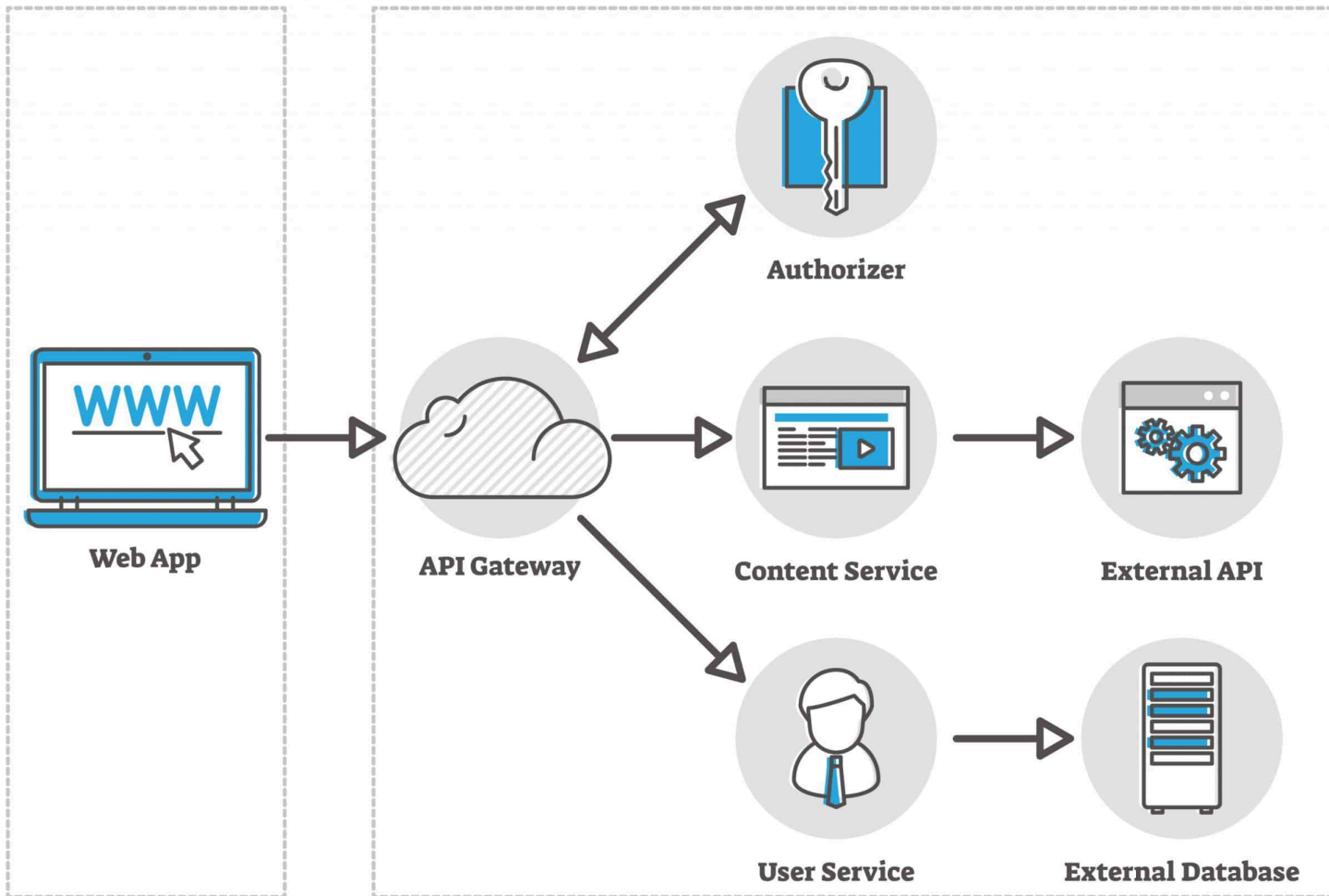
Microservices é uma arquitetura de software que consiste em um conjunto de serviços independentes, cada um com sua própria lógica de negócio e tecnologia.

## Monolitos Modulares

O monolito modular é uma arquitetura que procura reduzir a complexidade dos microserviços, porém garantindo escalabilidade e flexibilidade.



# SERVERLESS

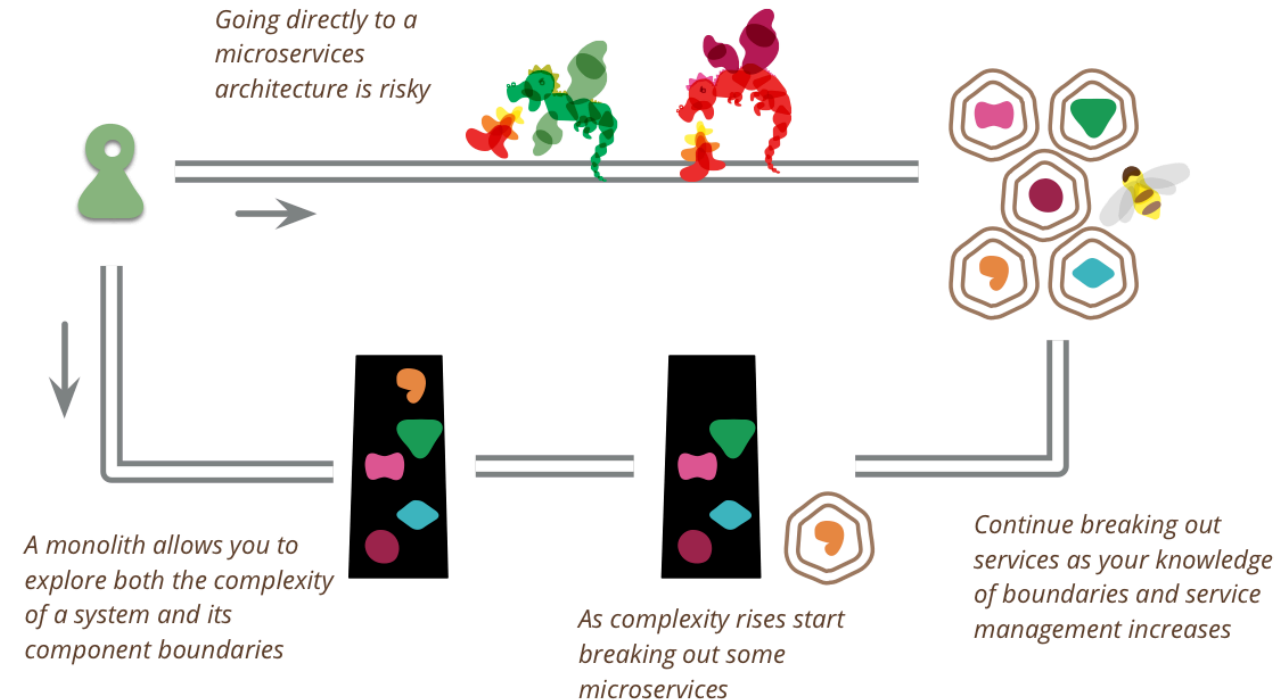


## Serverless

Em uma arquitetura Serverless, a infraestrutura é gerenciada pelo provedor de serviços e o desenvolvedor se concentra apenas na lógica de negócio.

## Bônus: Monolith First

*Monolith First* é uma abordagem descrita por *Martin Fowler* que consiste em iniciar o desenvolvimento de um sistema monolítico e, à medida que o sistema cresce, identificar pontos de acoplamento e refatorar para uma arquitetura mais flexível, como microserviços.



## Porque existem tantas arquiteturas?





## Contextos diferentes

Softwares variam muito em **escopo, complexidade, tempo de vida, restrições técnicas e equipe envolvida.**

Cada padrão de arquitetura surgiu como resposta a problemas específicos nesses contextos.

## Evolução do Conhecimento

A engenharia de software é uma disciplina jovem. À medida que aprendemos com erros do passado (excesso de acoplamento, baixa testabilidade, dificuldade de manutenção), surgem novas propostas para tratar melhor esses problemas.

## Focos Distintos

Cada arquitetura costuma enfatizar uma ou mais dimensões:

- Organização de código (Onion, Hexagonal)
- Separação de responsab. (MVC)
- Modelagem do domínio (DDD)
- Experiência de desenvolvimento e legibilidade (Screaming)
- Fluxo de execução e comportamento (Event Driven)

## Podemos usar mais de uma arquitetura?

Sim - e frequentemente fazemos isso, mas de forma seletiva e contextualizada.  
**Nenhuma arquitetura é uma bala de prata.**

Exemplo:

- DDD para modelar bem o domínio;
- Arquitetura Limpa ou Hexagonal para estruturar camadas e dependências;
- MVC no front-end;
- BCE como guia de organização por tipo de classe;
- Screaming Architecture na estrutura de diretórios.
- Monolito modular para escalabilidade e flexibilidade.

# Materiais de Apoio

- [Clean Architecture](#)
- [Software Architecture Guide](#)
- [Software Architecture Patterns](#)
- [Understand Clean Architecture in 7 Minutes](#)
- [Top 5 Most Used Architecture Patterns](#)
- [Descomplicando Clean Architecture - O que é a Arquitetura Limpa?](#)
- [SOLID, Clean Code e Design Patterns: Pare de Buscar o Código Perfeito!](#)
- [Monolith First](#)