

# Contextualizando o Desenvolvimento Web com Spring Boot 3 e Kotlin

**Camila Cavalcante**

Tech Education Coordinator  
DIO

 [linkedin.com/in/cami-la](https://www.linkedin.com/in/cami-la)

 [github.com/cami-la](https://github.com/cami-la)

 [instagram.com/camimi\\_la](https://www.instagram.com/camimi_la)

# Objetivo Geral

Para o desenvolvimento de uma aplicação web, é importante utilizar ferramentas modernas e confiáveis para garantir a qualidade, desempenho e segurança de um software.

Neste curso, vamos conhecer algumas ferramentas pertinentes para o desenvolvimento de um produto computacional de qualidade, uma **Rest API** Spring Boot e Kotlin.



# Pré-Requisitos

- IDE para desenvolvimento Kotlin (IntelliJ Community)
- JDK 17+
- Kotlin 1.7.22
- Sintaxe básica Kotlin
- Conhecimento acerca de POO



# Percurso

## Parte 1

Entendendo a Arquitetura Rest

## Parte 2

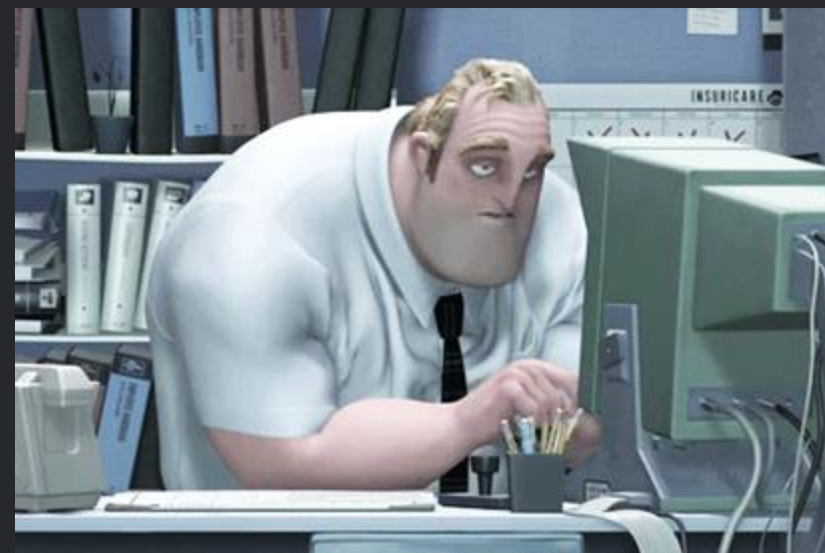
Overview do Spring Framework

## Parte 3

Arquitetura de Três Camadas com Spring Boot

# Dúvidas?

- > GitHub
- > Comunidade Online (Rooms)
- > Fórum do Bootcamp e/ou Artigos
- > Central de Ajuda DIO





## Parte 1



# Entendendo a Arquitetura Rest

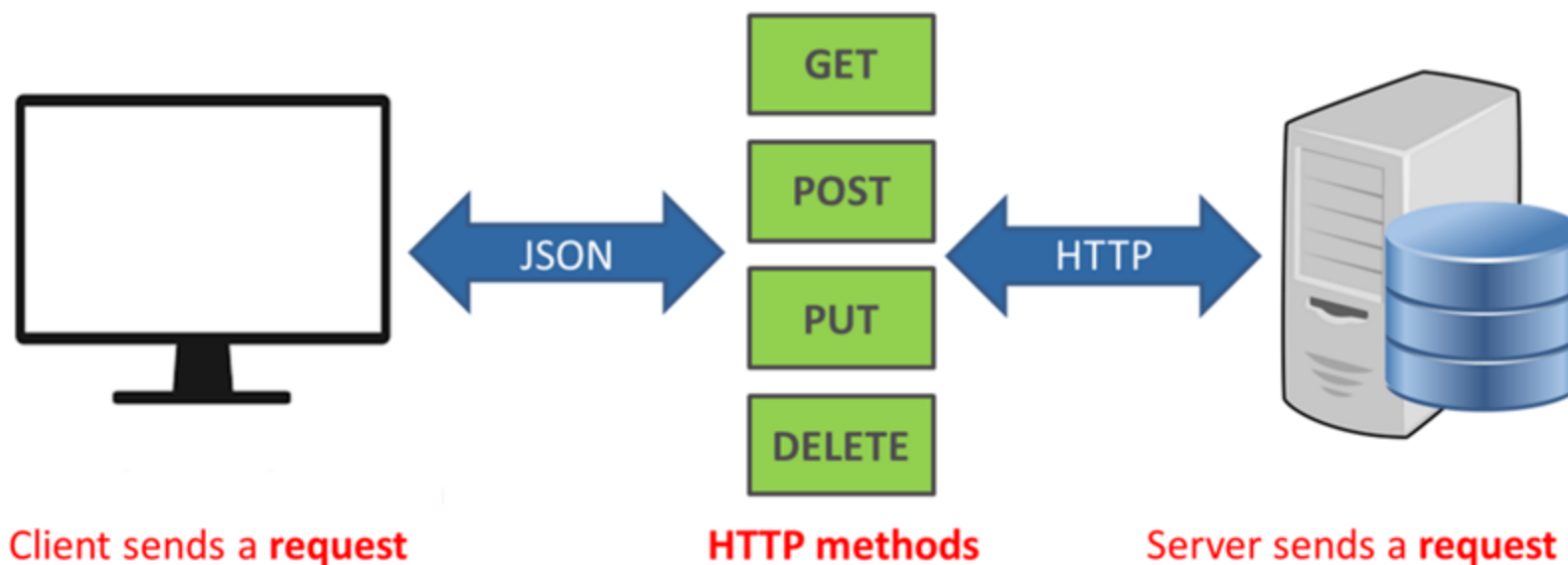
// Contextualizando o Desenvolvimento Web com Spring  
Boot 3 e Kotlin

# O que é API?

- API significa Application Programming Interface
- No contexto de APIs, a palavra **Aplicação** refere-se a qualquer software com uma função distinta.
- A **Interface** pode ser pensada como um *contrato de serviço* entre duas aplicações.
- Esse contrato define como as duas se comunicam usando solicitações e respostas.
- A documentação de suas respectivas APIs contém informações sobre como os desenvolvedores devem estruturar essas solicitações e respostas.

# Como as APIs funcionam?

- A arquitetura da API geralmente é explicada em termos de cliente e servidor.
- A aplicação que envia a solicitação é chamada de **cliente** e a aplicação que envia a resposta é chamada de **servidor**.





# Como as APIs funcionam?

- **APIs SOAP:** Cliente e servidor trocam mensagens usando XML. Esta é uma API menos flexível que era mais popular.
- **APIs RPC:** O cliente conclui uma função (ou um procedimento) no servidor e o servidor envia a saída de volta ao cliente.
- **APIs WebSocket:** O servidor pode enviar mensagens de retorno de chamada a clientes conectados, tornando-o mais eficiente que a API REST.
- **APIs REST:** O cliente envia solicitações ao servidor como dados. O servidor usa essa entrada do cliente para iniciar funções internas e retorna os dados de saída ao cliente.

# O que são APIs REST?

- REST significa Transferência Representacional de Estado.
- Clientes e servidores trocam dados usando [HTTP](#).
- O HTTP permite criar, atualizar, pesquisar, executar e remover operações, atuando sob determinados recursos.
- A principal característica da API REST é a ausência de estado.



# Métodos e Status HTTP

SAFE METHODS NO ACTION ON SERVER	{	GET	HTTP/1.1 MUST IMPLEMENT THIS METHOD
MESSAGE WITH BODY		HEAD	INSPECT RESOURCE HEADERS
SEND DATA TO SERVER	{	PUT	DEPOSIT DATA ON SERVER — INVERSE OF GET
		POST	SEND INPUT DATA FOR PROCESSING
		PATCH	PARTIALLY MODIFY A RESOURCE
		TRACE	ECHO BACK RECEIVED MESSAGE
		OPTIONS	SERVER CAPABILITIES
		DELETE	DELETE A RESOURCE — NOT GUARANTEED

## HTTP STATUS CODES

### 2xx Success

200 Success / OK

### 3xx Redirection

301 Permanent Redirect

302 Temporary Redirect

304 Not Modified

### 4xx Client Error

401 Unauthorized Error

403 Forbidden

404 Not Found

405 Method Not Allowed

### 5xx Server Error

501 Not Implemented

502 Bad Gateway

503 Service Unavailable

504 Gateway Timeout

freemove

# JSON

- O JSON é um formato de troca de dados entre sistemas independente de linguagem de programação derivado do JavaScript.
- É frequentemente utilizado em aplicações Ajax, configurações, bancos de dados e serviços web RESTful.



Diagram illustrating a RESTful API request and response using JSON.

The request is a **POST** to the endpoint `http://localhost:8080/sign-up`.

The request headers include **Content-Type: application/json**.

The request body (JSON) is:

```
1 {
2   "firstName": "Camila",
3   "lastName": "Cavalcante",
4   "password": "1234",
5   "email": "cami@email.com"
6 }
```

# Referências

- O que é uma API?
- Arquitetura REST: Saiba o que é e seus diferenciais
- Introdução ao JSON: Um Guia Para JSON que vai Direto ao Ponto
- Design de APIs RESTful (Melhores Práticas)





## Parte 2



# Overview Spring Framework

// Contextualizando o Desenvolvimento Web com Spring  
Boot 3 e Kotlin

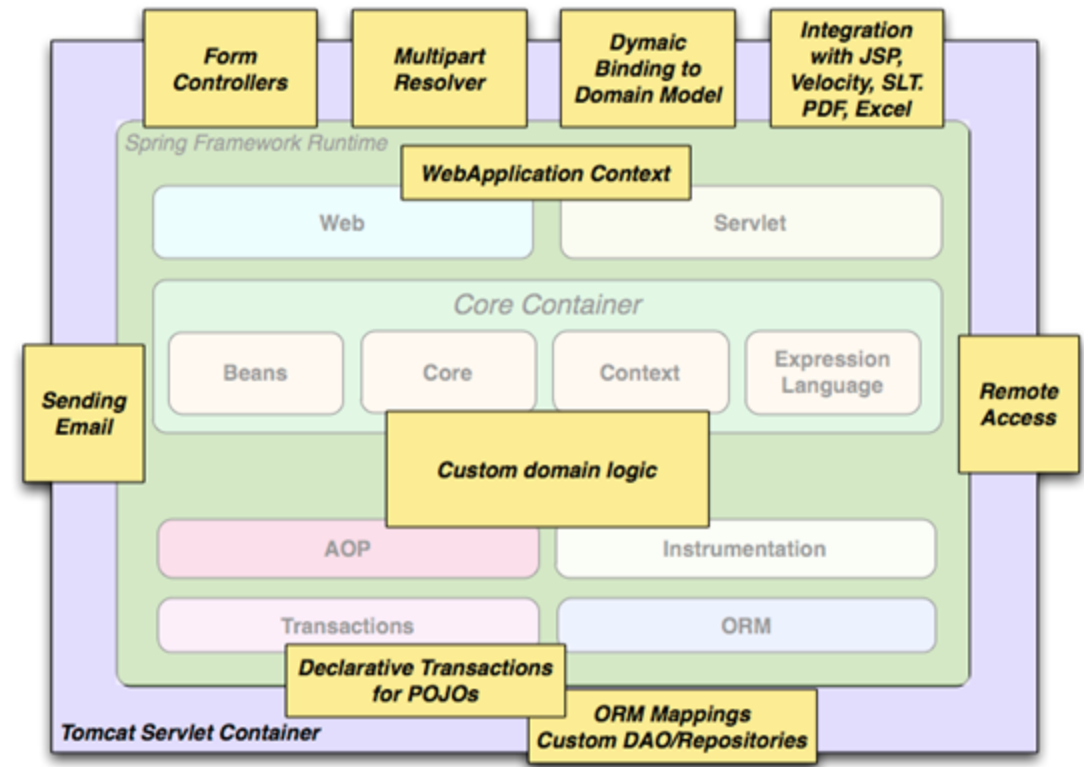
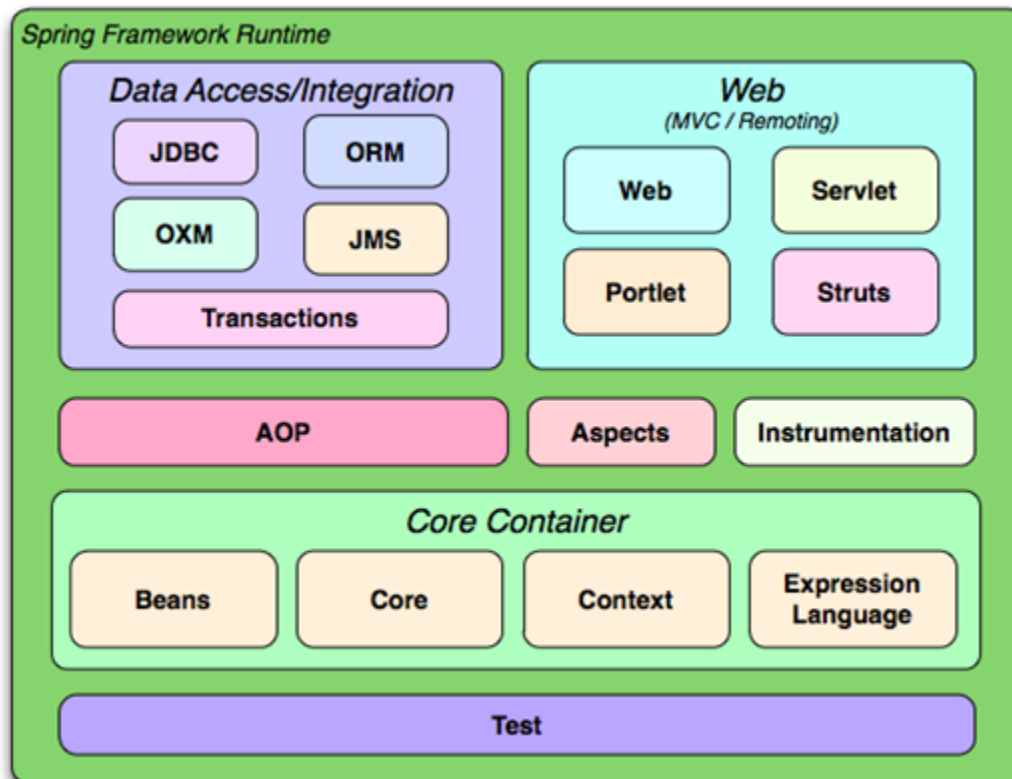
# O que é o Spring Framework?

- O Spring é um framework Java criado com o objetivo de facilitar o desenvolvimento de aplicações.
- Baseado na IoC e DI, fornecendo para isso um container, que representa o núcleo do framework e que é responsável por criar e gerenciar os componentes da aplicação, os quais são comumente chamados de beans.
- [Spring Boot](#) é um framework Java open source ele traz mais agilidade para o processo de desenvolvimento, uma vez que devs conseguem reduzir o tempo gasto com as configurações iniciais.



# O que é o Spring Framework

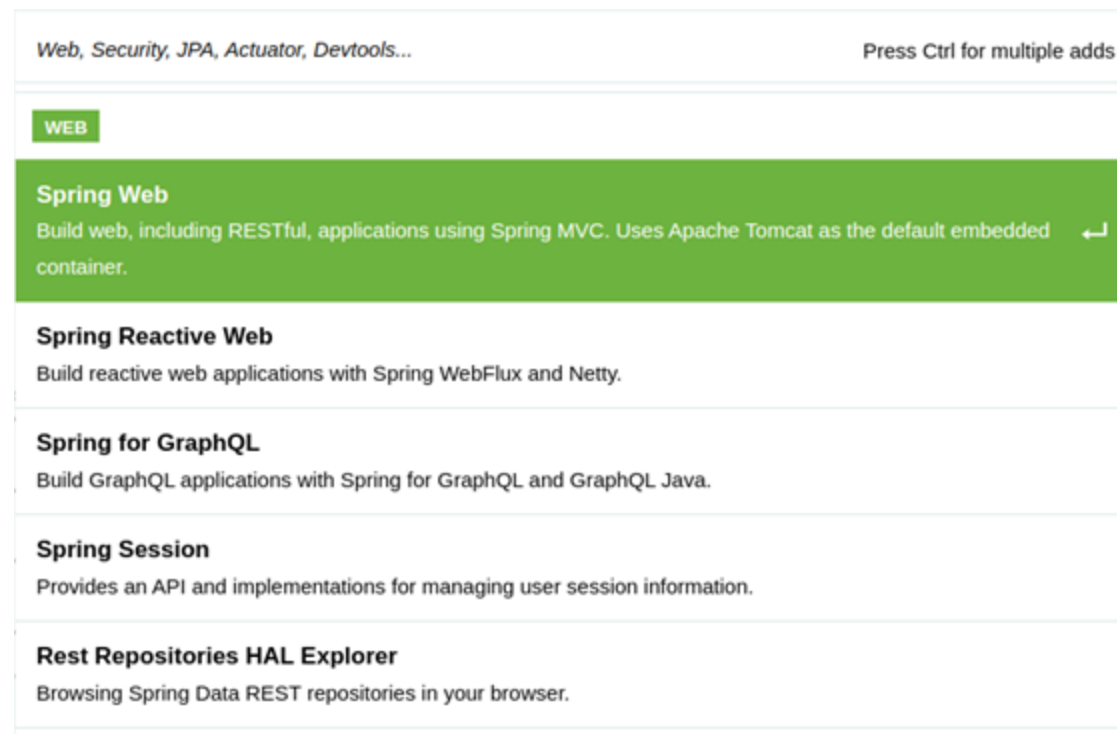
- O Spring Framework consiste em recursos organizados em cerca de 20 módulos.





# Spring Boot Starters

- Com o Spring Boot conseguimos abstrair e facilitar a configuração de, por exemplo:
  - Servidores;
  - Gerenciamento de dependências;
  - Configurações de bibliotecas;
  - Métricas & health checks;
  - Entre outros!



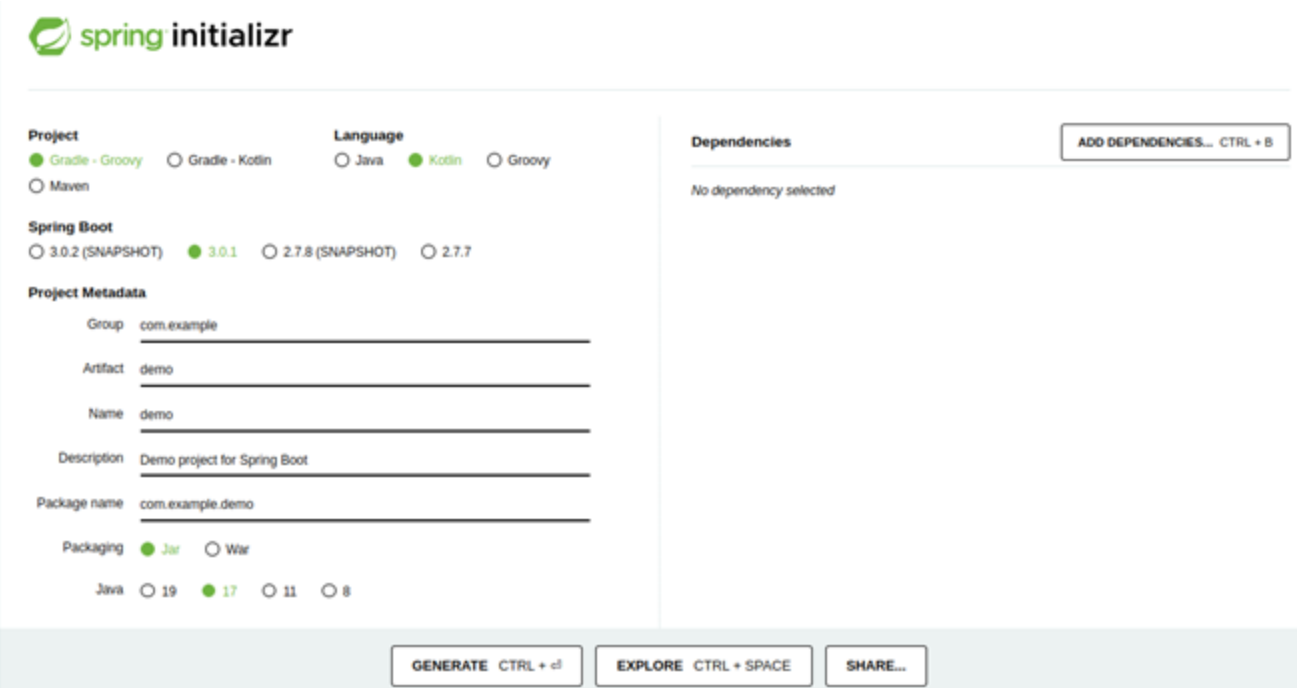
# Spring Boot Starters

- Os starters são dependências que agrupam outras dependências com um propósito em comum. Dessa forma, somente uma configuração é realizada no seu gerenciador de dependências.
- Por exemplo, o `spring-boot-starter-test`, contém funcionalidades úteis e anotações que facilitam e ajudam a testar sua aplicação.

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
}
```

# Spring Initializr

- Para facilitar a criação de aplicações utilizando outras IDEs o Spring disponibilizou o [Spring Initializr](#). Ele é uma UI que permite a criação de projetos Sprint Boot de forma facilitada.



The screenshot displays the Spring Initializr web interface. At the top left is the 'spring initializr' logo. The main configuration area is divided into several sections: 'Project' with radio buttons for 'Gradle - Groovy' (selected), 'Gradle - Kotlin', 'Maven', 'Java' (selected), 'Kotlin', and 'Groovy'; 'Spring Boot' with radio buttons for '3.0.2 (SNAPSHOT)', '3.0.1' (selected), '2.7.8 (SNAPSHOT)', and '2.7.7'; 'Project Metadata' with text input fields for 'Group' (com.example), 'Artifact' (demo), 'Name' (demo), 'Description' (Demo project for Spring Boot), and 'Package name' (com.example.demo); and 'Packaging' with radio buttons for 'Jar' (selected) and 'War'. Below these are radio buttons for 'Java' versions: '19', '17' (selected), '11', and '8'. On the right, the 'Dependencies' section shows 'No dependency selected' and a button 'ADD DEPENDENCIES... CTRL + B'. At the bottom, there are three buttons: 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE', and 'SHARE...'.

# Referências

- [Spring Boot: como começar](#)
- [Introduction to Spring Framework](#)



## Parte 3



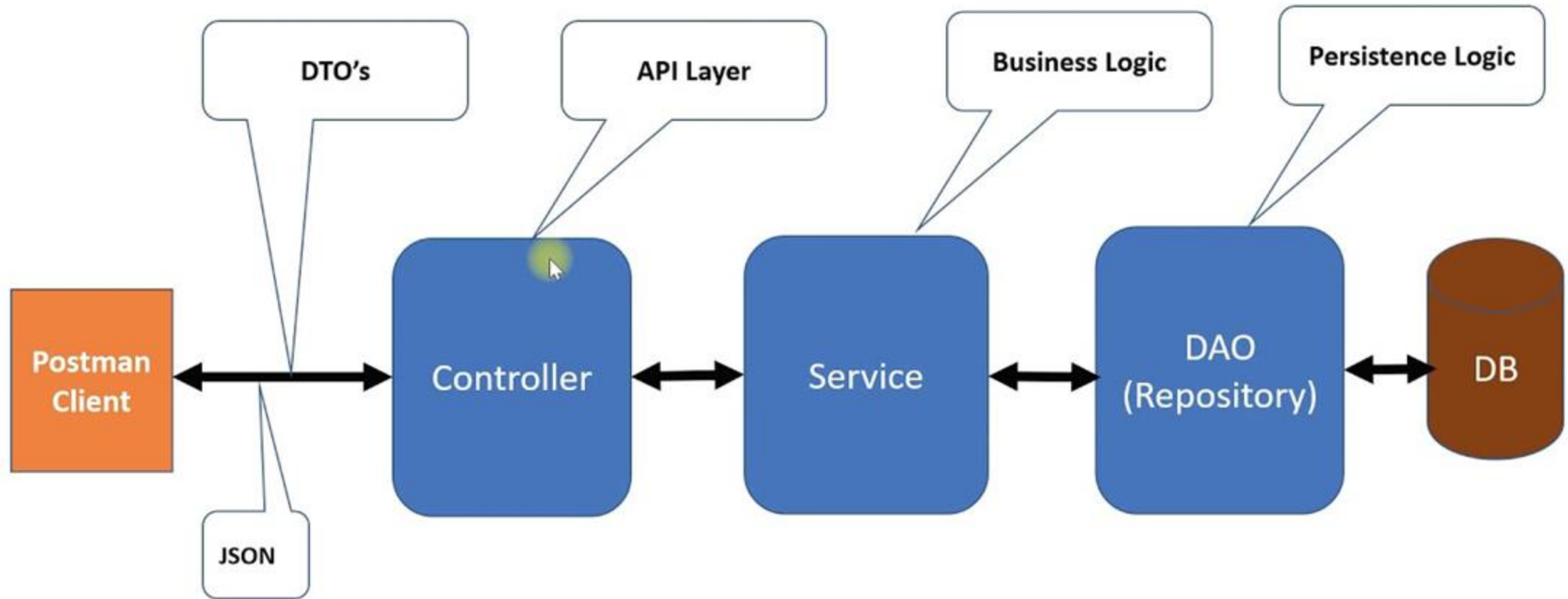
# Arquiteturas de três camadas

// Contextualizando o Desenvolvimento Web com Spring Boot 3 e Kotlin

# Arquitetura de três camadas

- A arquitetura em três camadas tem por objetivo promover a separação das funcionalidades usando camadas para a separação da lógica de apresentação, lógica de negócio e lógica de acesso a dados.
- A separação em três camadas torna o sistema mais flexível, permitindo que as camadas sejam desenvolvidas e modificadas independentemente.

# Arquitetura Projeto Spring



# Arquivo de Configuração

- Ao trabalhar com Spring Boot, nos deparamos com várias configurações que devem ser realizadas.
- O Spring Boot permite utilizar 2 diferentes tipos de arquivos de configurações: **application.properties** ou **application.yml**

```
3 # https://www.baeldung.com/spring-boot-h2-database#database-configuration
4 spring.datasource.url=jdbc:h2:mem:dio-game-awards
5 spring.datasource.driverClassName=org.h2.Driver
6 spring.datasource.username=sa
7 spring.datasource.password=password
8
9 # https://www.baeldung.com/spring-boot-h2-database#h2-console
10 spring.h2.console.enabled=true
11 spring.h2.console.path=/h2-console
12 spring.h2.console.settings.trace=false
13 spring.h2.console.settings.web-allow-others=false
14
15 # https://www.baeldung.com/spring-boot-data-sql-and-schema-sql#theschemasqlfile
16 spring.jpa.defer-datasource-initialization=true
```

```
1 #---- H2 Database ----
2 spring:
3   datasource:
4     url: jdbc:h2:mem:credit_request_system_db
5     username: sa
6     password:
7   jpa:
8     show-sql: true
9     hibernate:
10       properties:
11         hibernate.format_sql: true
12   h2:
13     console:
14       enabled: true
15       path: /h2-console
16       settings:
17         trace: false
18         web-allow-others: false
```



# Referências

- Arquitetura em 3 Camadas
- Spring Initializr
- Common Application Properties



# Dúvidas?

- > GitHub
- > Comunidade Online (Rooms)
- > Fórum do Bootcamp e/ou Artigos
- > Central de Ajuda DIO

