

Palestra

Java Spring Boot

Conheça a Tecnologia
Back-End Amplamente
Utilizada no Mercado

Prof. Antonio B. C. Sampaio Jr



25 de Setembro – 8h às 9h

PROF. SAMPAIO

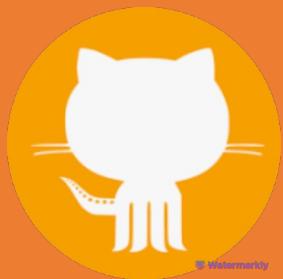


- Trabalha na Área de Tecnologia e Educação há mais de 25 anos.
- Já capacitou mais de 10.000 alunos no formato presencial e mais de 10.000 alunos no formato EAD.
- Mestre em Informática pela PUC-RIO.
- Certificado JAVA.
- Certificado JS e REACT.
- Atua na Área de TI na Receita Federal do Brasil.

www.amazoncode.com.br

CEL/ZAP: (91) 98216-1883

REPOSITÓRIO GITHUB



[antonio-sampaio-jr / palestra-stefanini](#)

TÓPICOS DA PALESTRA

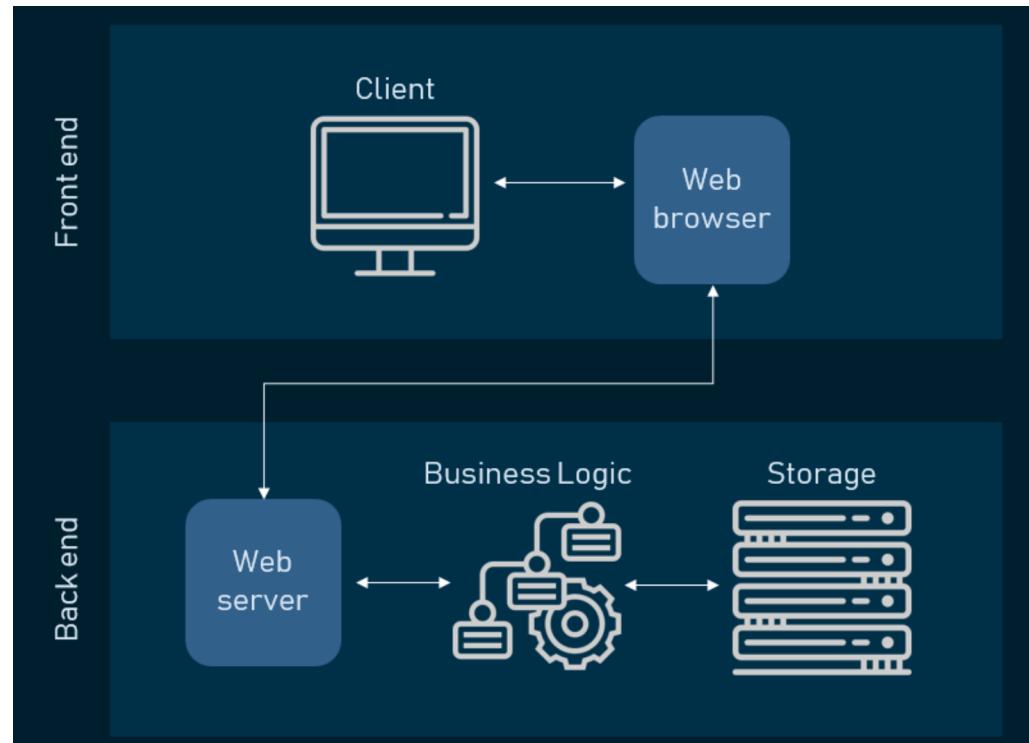


- CONCEITOS BÁSICOS
- API
- WEB SERVICES
- REST API
- VISÃO GERAL DO SPRING FRAMEWORK
- VISÃO GERAL DO SPRING BOOT
- ESTRUTURA DE UM PROJETO SPRING BOOT
- PADRÕES DE PROJETO IOC & DI
- ARQUITETURA SPRING BOOT
- PROJETO PRÁTICO

CONCEITOS BÁSICOS

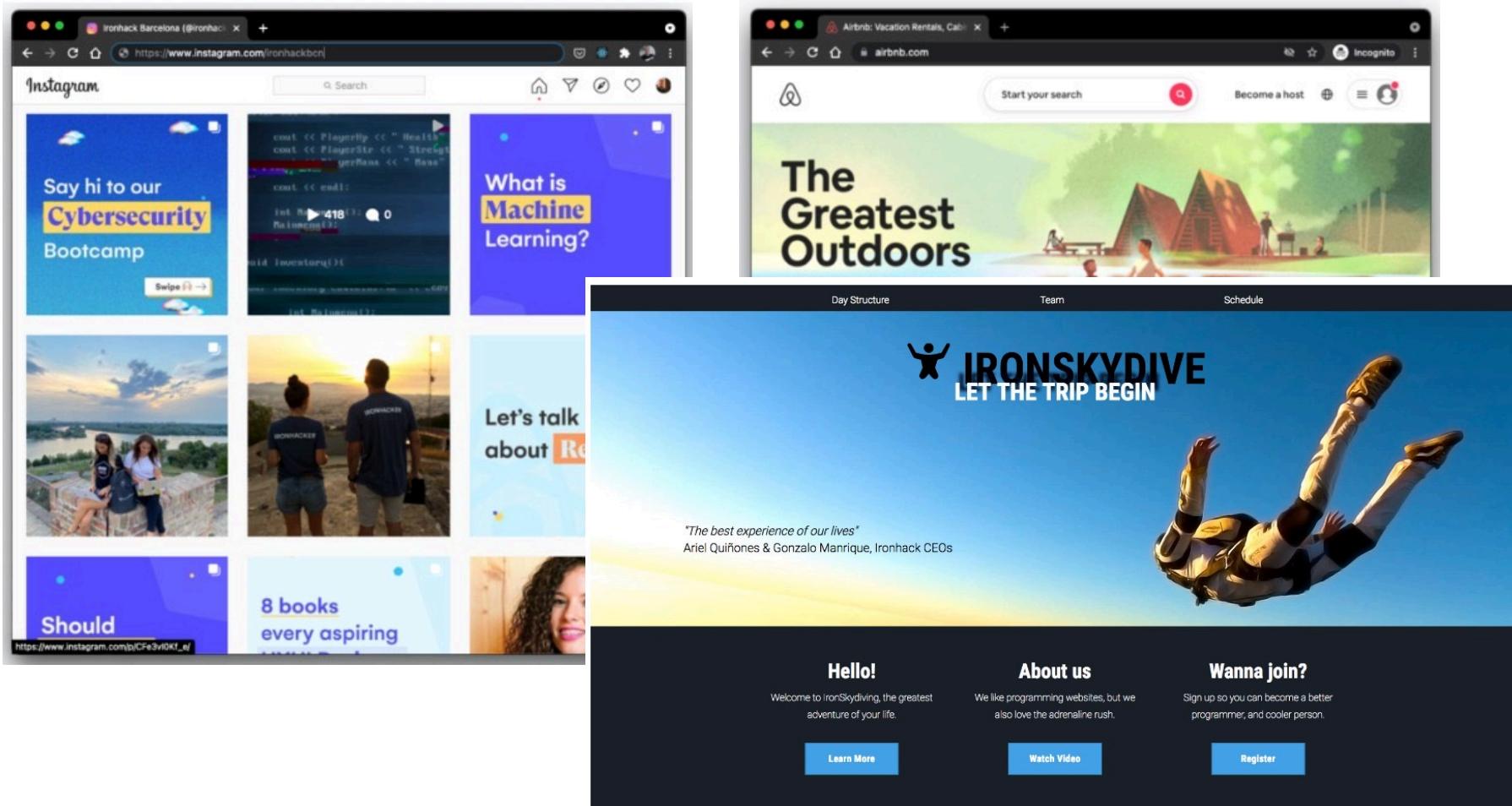
FRONT-END & BACK-END

- **O que é Front-end?**
 - É toda a parte da programação relativa à interface de uma aplicação (Telas).
- **O Que é Back-end?**
 - É toda a parte da programação relativa às implementações das regras de negócios de uma aplicação.
- É o “coração” da aplicação.



<https://www.tabnews.com.br/maxdev/interessado-em-aprender-a-programar-entenda-o-que-e-front-end-back-end-e-mobile>

EXEMPLO DE FRONT-END

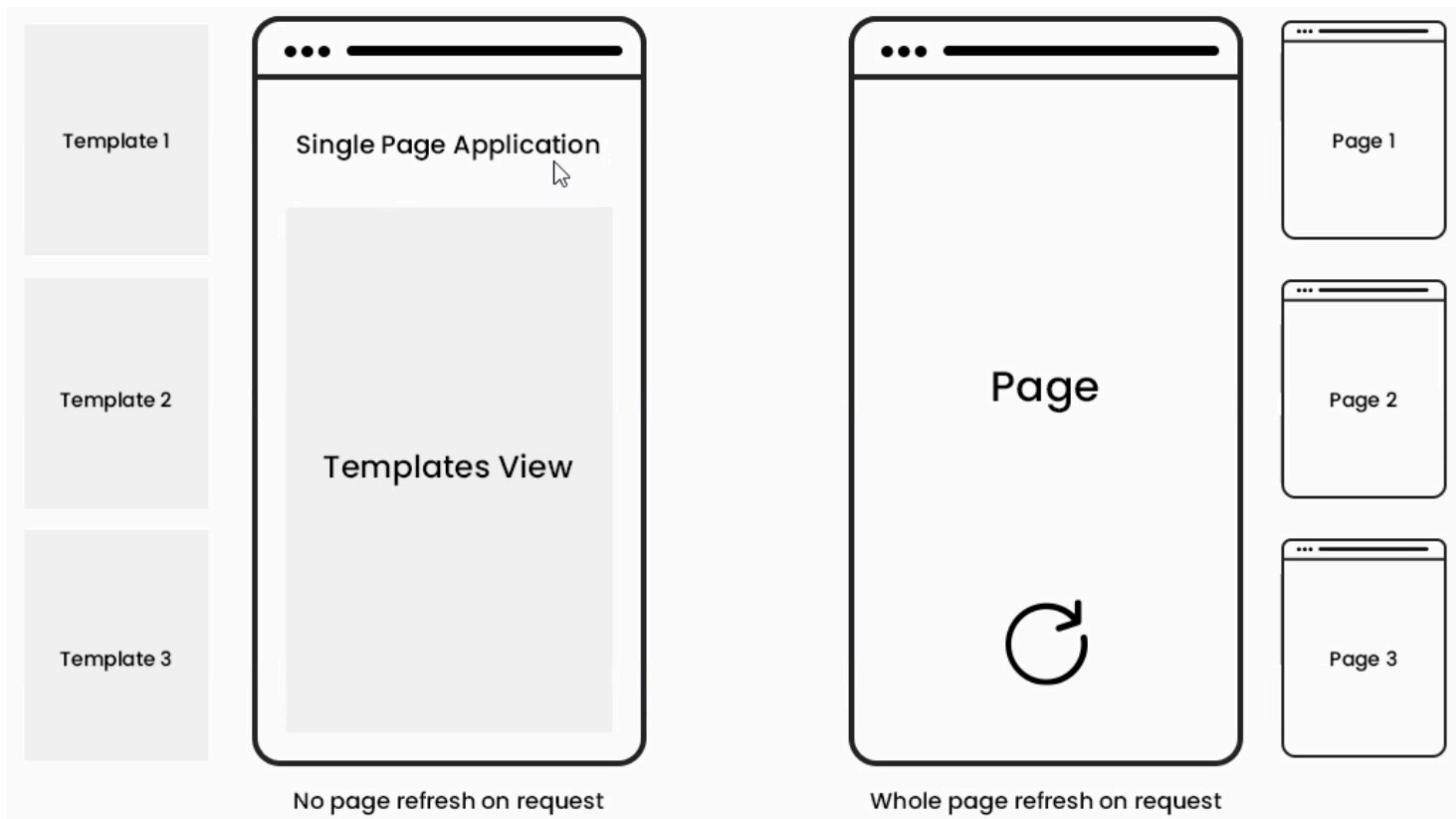


FRAMEWORKS FRONT-END

- Principais Frameworks de Front-end?
 - **React JS (Facebook)** – Biblioteca de JavaScript declarativa, eficiente e flexível para a criação de interfaces do usuário (UI).
 - **Angular JS (Google)** – Framework JavaScript ideal para a criação de aplicações Web baseadas na filosofia SPA (*Single Page Application*).
 - **Vue.js** – Framework open source JavaScript baseado em SPA e utilizado para a criação de interfaces interativas.
 - **Bootstrap** – Framework CSS que possibilita a criação de páginas Web bonitas e funcionais.

FRAMEWORKS FRONT-END

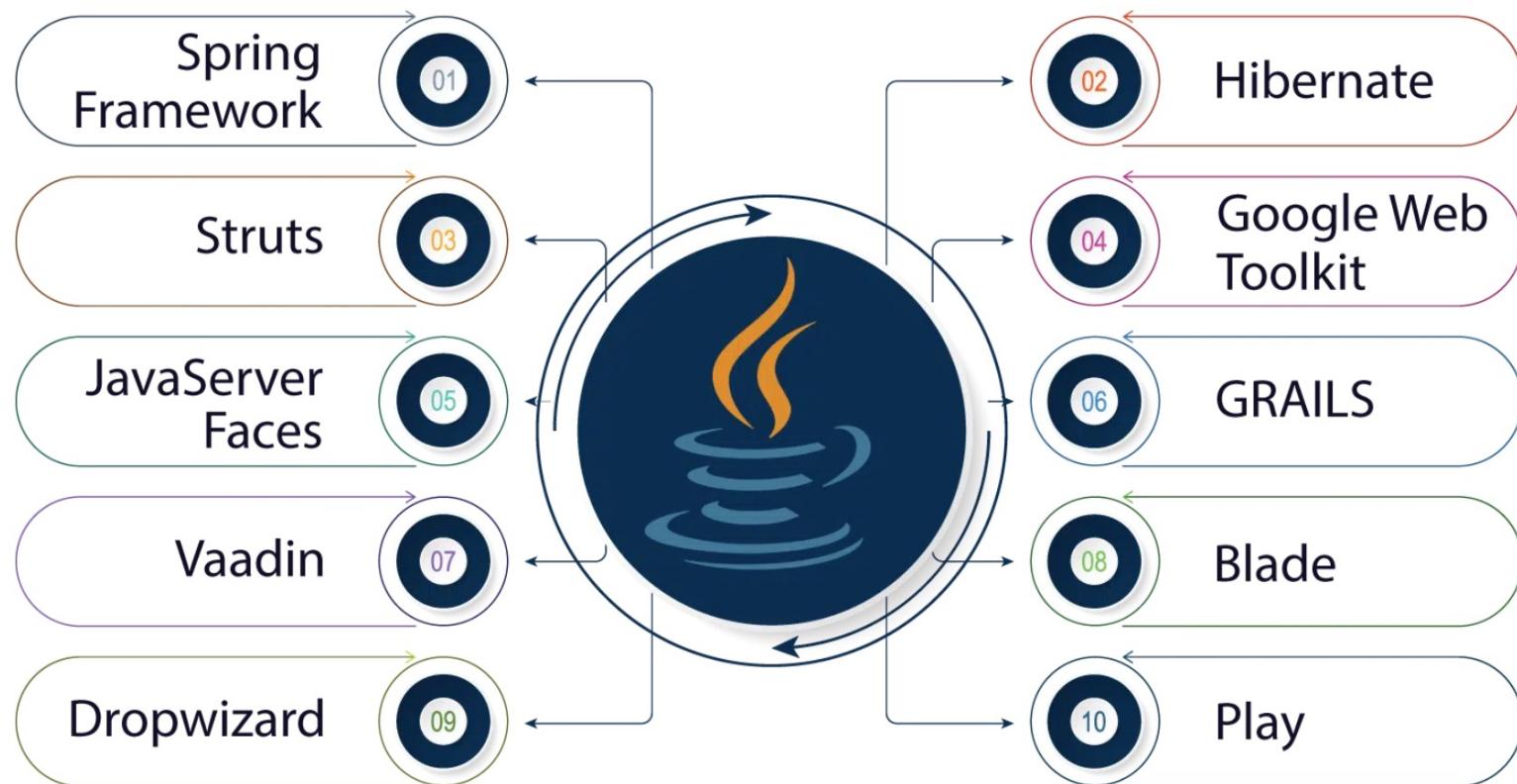
- O Que é a Filosofia SPA (Single Page Application)?



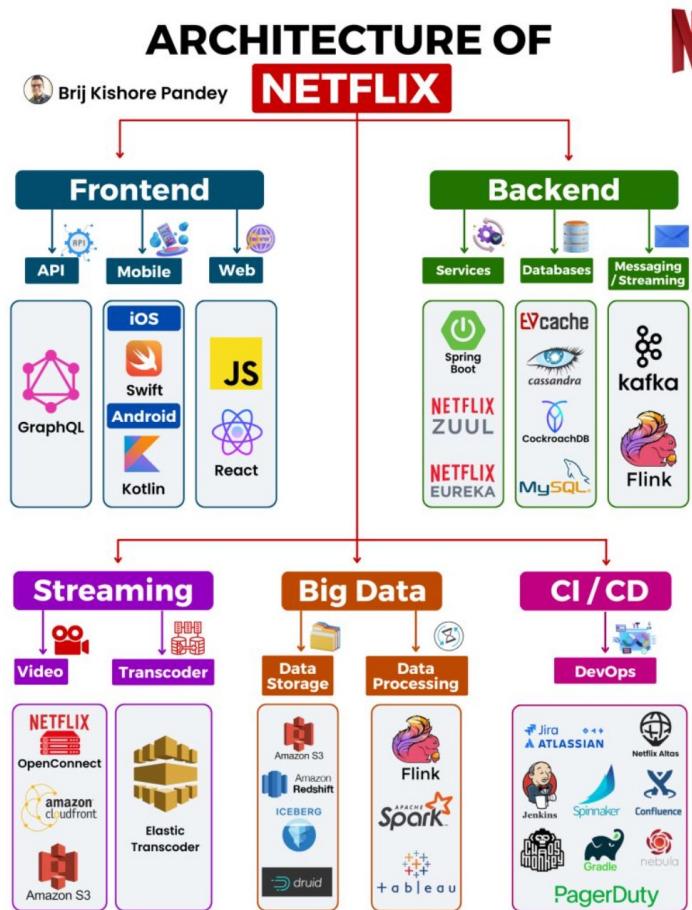
FRAMEWORKS BACK-END

- Principais Frameworks de Back-end?
 - **Spring Boot (JAVA)** - Utilizado no Trivago, Via Varejo, etc.
 - **Django (Python)** – Utilizado no Instagram, Coursera, etc.
 - **Ruby on Rails (Ruby)** – Utilizado no ZendDesk, GitHub, etc.
 - **Express JS (JavaScript)** – Utilizado no MySpace, Storify, etc.
 - **CakePHP (PHP)** - Utilizado no Mapme, Followmy TVy, etc.

FRAMEWORKS JAVA



TECNOLOGIAS NETFLIX



API

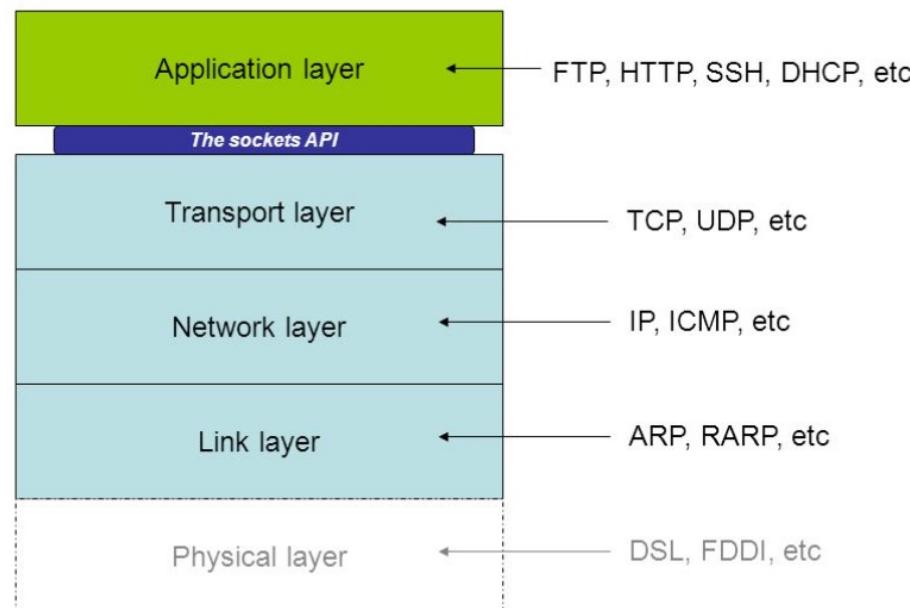
API

- **O que é API?**

- API (*Application Program Interface*) é um conjunto de definições e protocolos que permitem a comunicação entre diferentes componentes de software. Em outras palavras, uma API é uma interface que define como as diferentes partes de um sistema de software interagem umas com as outras.
- Uma API pode ser utilizada para muitas finalidades diferentes, como permitir que diferentes sistemas ou aplicativos se comuniquem, fornecer acesso programático a recursos e serviços em uma plataforma, definir um conjunto de operações padronizadas que podem ser usadas por desenvolvedores para interagir com um sistema, entre outras.

API

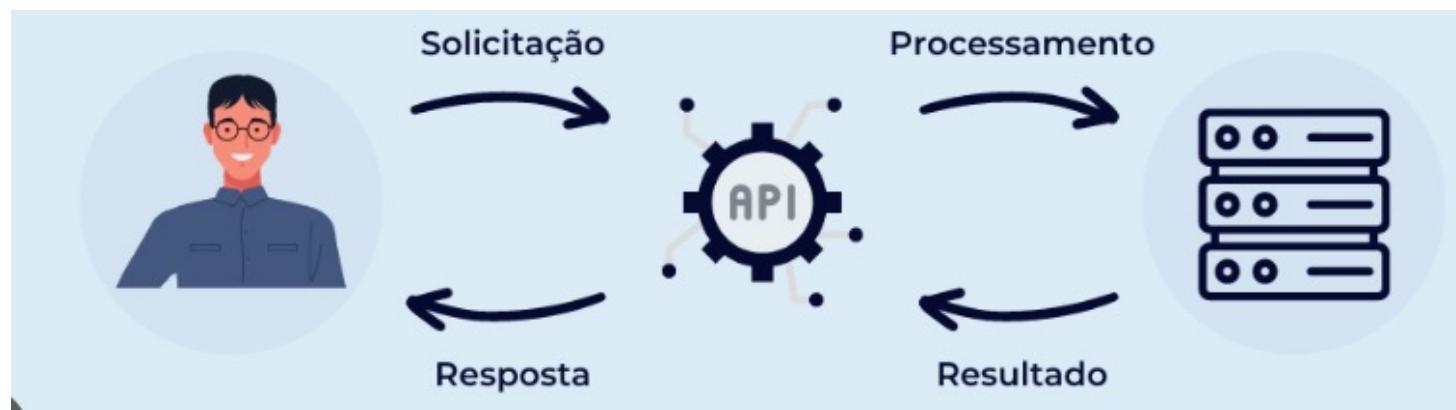
- Arquitetura TCP/IP



- A interface sockets API pode ser pensada como um contrato de serviço entre as duas camadas: **Aplicação e Transporte**. Esse contrato define como essas duas camadas se comunicam trocando solicitações e respostas.

TIPOS DE APIs

- Existem quatro maneiras diferentes pelas quais as APIs podem funcionar:
 - (1) APIs baseadas em solicitação e resposta: Neste tipo de API, o cliente faz uma solicitação para o servidor através de uma chamada de API e o servidor retorna uma resposta. As APIs baseadas em solicitação e resposta podem usar vários protocolos, como **HTTP/REST**, SOAP/XML, gRPC/protobuf, entre outros.



<https://mambowifi.com/api-x-webhook-qual-a-diferenca-e-em-quais-casos-usar/>

TIPOS DE APIs

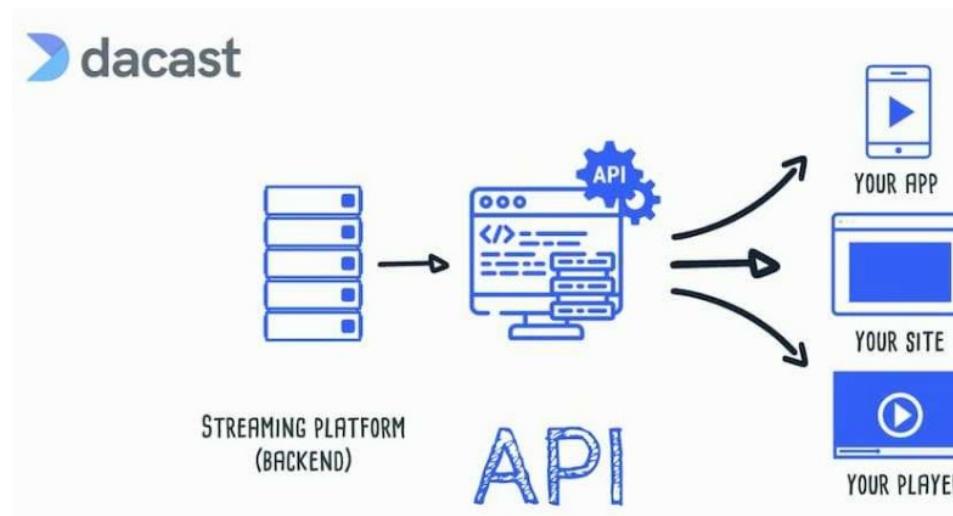
- Existem quatro maneiras diferentes pelas quais as APIs podem funcionar:
 - (1) APIs baseadas em solicitação e resposta:
HTTP/REST



<https://research.aimultiple.com/graphql-vs-rest/>

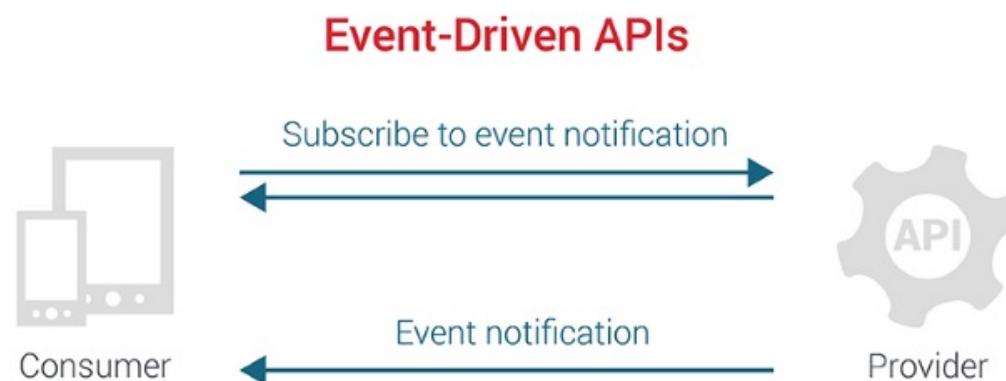
TIPOS DE APIs

- Existem quatro maneiras diferentes pelas quais as APIs podem funcionar:
 - (2) APIs baseadas em streaming: Neste tipo de API, o cliente pode enviar ou receber dados em tempo real por meio de um fluxo contínuo de dados. As APIs baseadas em streaming são usadas em aplicativos que exigem comunicação em tempo real, como bate-papo em tempo real, transmissão de vídeo, jogos on-line, entre outros.



TIPOS DE APIs

- Existem quatro maneiras diferentes pelas quais as APIs podem funcionar:
 - (3) APIs baseadas em eventos: Neste tipo de API, o servidor envia uma mensagem ao cliente sempre que um evento ocorre. As APIs baseadas em eventos são usadas para notificar os clientes de alterações em dados, como novas mensagens em um fórum de discussão ou um novo pedido em um sistema de comércio eletrônico.



<https://blog.axway.com/learning-center/apis/api-streaming/event-driven-vs-rest-api-interactions>

TIPOS DE APIs

- Existem quatro maneiras diferentes pelas quais as APIs podem funcionar:
 - (4) APIs baseadas em consulta: Neste tipo de API, o cliente pode enviar consultas complexas para o servidor, que retorna os resultados da consulta. As APIs baseadas em consulta são usadas para buscar informações de um banco de dados ou para realizar operações em dados existentes, como atualizar ou excluir registros.



GRAPHQL QUERY

```
{  
  person {  
    firstName  
    lastName  
  }  
}
```

GRAPHQL JSON

```
{  
  "data": {  
    "person": {  
      "firstName": "John",  
      "lastName": "Smith",  
    }  
  }  
}
```

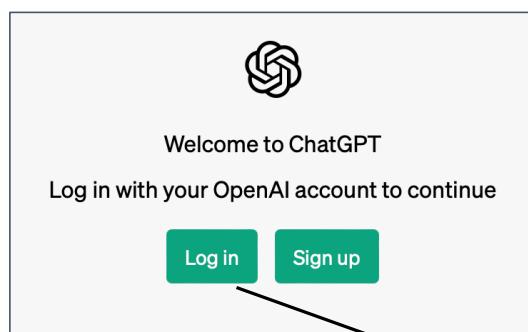
<https://research.aimultiple.com/graphql-vs-rest/>

ESCOPO DAS APIs

- São 04 os Escopos de Uso
 - APIs Privadas
 - Elas são internas a uma empresa e são usadas apenas para conectar sistemas e dados dentro da empresa.
 - APIs Públicas
 - Estas são abertas ao público e podem ser usadas por qualquer pessoa. Pode ou não haver alguma autorização e custo associado a esses tipos de APIs.
 - APIs de Parceiros
 - Estas são acessíveis apenas por desenvolvedores externos autorizados para auxiliar as parcerias entre empresas.
 - APIs Compostas
 - Estas combinam duas ou mais APIs distintas para atender a requisitos ou comportamentos complexos do sistema.

ESCOPO DAS APIs

- APIs de Parceiros



Welcome back

Email address

Continue

Don't have an account? [Sign up](#)

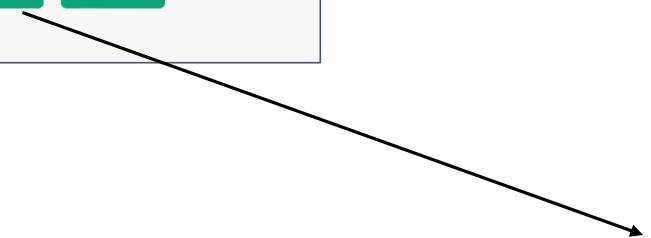
OR

Continue with Google

Continue with Microsoft Account

API GOOGLE

API
MICROSOFT



ESCOPO DAS APIs

- APIs Pùblicas

The screenshot shows the Twitter Developer Platform interface. On the left, there's a sidebar with links like 'Getting started', 'Tools and libraries', 'What to build', 'Migrate' (which is expanded), 'Overview', 'Data format migration', 'Twitter API endpoint map' (which is selected and highlighted in dark blue), 'Twitter API v2', and 'Enterprise - Gnip 2.0'. At the top, there are navigation tabs: 'Products', 'Docs' (which is underlined in blue), 'Use Cases', 'Community', and 'Support'. The main content area has two columns. The left column is titled 'Tweets' and lists several API endpoints: 'GET statuses/show', 'GET statuses/lookup', 'POST statuses/update', 'POST statuses/destroy/:id', 'GET statuses/user_timeline', 'GET statuses/mentions_timeline', and 'GET statuses/home_timeline'. The right column is titled 'Standard v1.1' and 'Premium v1.1', showing the same list of endpoints.

<https://developer.twitter.com/en/docs/twitter-api/migrate/twitter-api-endpoint-map>

ESCOPO DAS APIs

- APIs PÚBLICAS

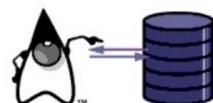
The screenshot shows the Meta for Developers documentation page for API Endpoints. The top navigation bar includes links for Documentos, Ferramentas, Suporte, and a search bar. The main content area is titled "API Endpoints" and "Nesta Página". It states that all endpoints start with <https://graph.facebook.com/v3.3>. Below this, a table lists four endpoints related to commerce orders:

Endpoint	Description
<code>GET /<page-id>/commerce_orders</code>	List all commerce orders associated with a page
<code>GET /<order-id></code>	Fetch order details for a given order ID
<code>GET /<order-id>/items</code>	Fetch line items for a given order ID
<code>GET /<order-id>/shipments</code>	Fetch all shipments for a given order ID

<https://developers.facebook.com/docs/commerce-platform/order-management/api-endpoints/>

APIs e Web Services

- Diferenças entre APIs e Web Services
 - APIs e Web Services são dois conceitos relacionados, mas diferentes. As APIs definem um conjunto de operações e funcionalidades que podem ser acessadas e utilizadas por outros programas. Não necessitam de comunicação via rede para funcionarem.
 - Exemplo: API JDBC



- Os Web Services são APIs que precisam enviar e receber dados pela rede.

APIs e Web Services

- **Diferenças entre APIs e Web Services**

- Webservices são um tipo de API que utilizam tecnologias da web para permitir a comunicação entre diferentes sistemas, plataformas e linguagens de programação. Existem diferentes tipos de Web Services, como SOAP/XML e REST/JSON, que utilizam diferentes protocolos e formatos de dados para a comunicação.
- Resumindo, enquanto APIs são interfaces de programação que podem ser utilizadas em diferentes tipos de aplicações, Web Services são um tipo de API que utiliza tecnologias da web para permitir a comunicação entre diferentes sistemas e plataformas.

WEB SERVICES

WEB SERVICES

- **Definição**
 - Os **Web Services** são componentes que permitem às aplicações enviar e receber dados em formato XML. Cada aplicação pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, o formato XML.
 - Principais padrões: **SOAP/XML** e **HTTP/JSON (REST)**.



WEB SERVICES

- Padrão SOAP/XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org
/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap
/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi
="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http
://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
    <ns1:MinhaUFLA.authResponse xmlns:ns1="urn:minhaufla">
        <return xsi:type="xsd:string">{&quot;id&quot;:&quot;800_1
        .4&quot;,&quot;message&quot;:&quot;ST-119242
        -BCv0xadIu9HBEBNZD6sk-casdgti&quot;,&quot;type&quot;
        ;:&quot;SUCESSO&quot;,&quot;system&quot;:&quot
        ;MINHAUFLA&quot;}</return>
    </ns1:MinhaUFLA.authResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Exemplo de mensagem SOAP-XML

WEB SERVICES

- Padrão HTTP/JSON (Padrão REST)

```
{  
    "id": "800_1.4",  
    "message": "131vnt2td8h1hhnaens1d7jla1",  
    "type": "SUCESSO",  
    "system": "MINHAUFLA"  
}
```

Exemplo de mensagem REST-JSON

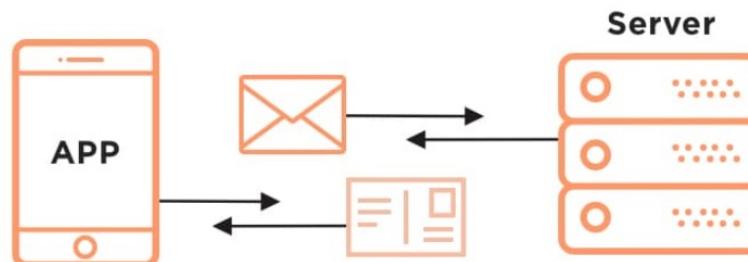
WEB SERVICES

- Comparação SOAP/XML & HTTP/JSON (REST)

SOAP vs. REST APIs

SOAP IS LIKE USING AN ENVELOPE

Extra overhead, more bandwidth required, more work on both ends(sealing and opening).



REST IS LIKE A POSTCARD

Lighterweight, can be cached, easier to update

Source: <https://assets-global.website-files.com>



FORMATO JSON

- JSON (*JavaScript Object Notation*) é um formato compacto, de padrão aberto independente, de troca de dados simples e rápida entre sistemas, especificado por Douglas Crockford em 2000, que utiliza texto legível a humanos, no formato atributo-valor.
- É ideal para enviar e receber informações pela Internet.

```
{  
    "id": 123,  
    "nome": "JSON T-Shirt",  
    "preco": 99.99,  
    "estoque": {  
        "deposito": 300,  
        "loja": 20  
    }  
}
```

REST API

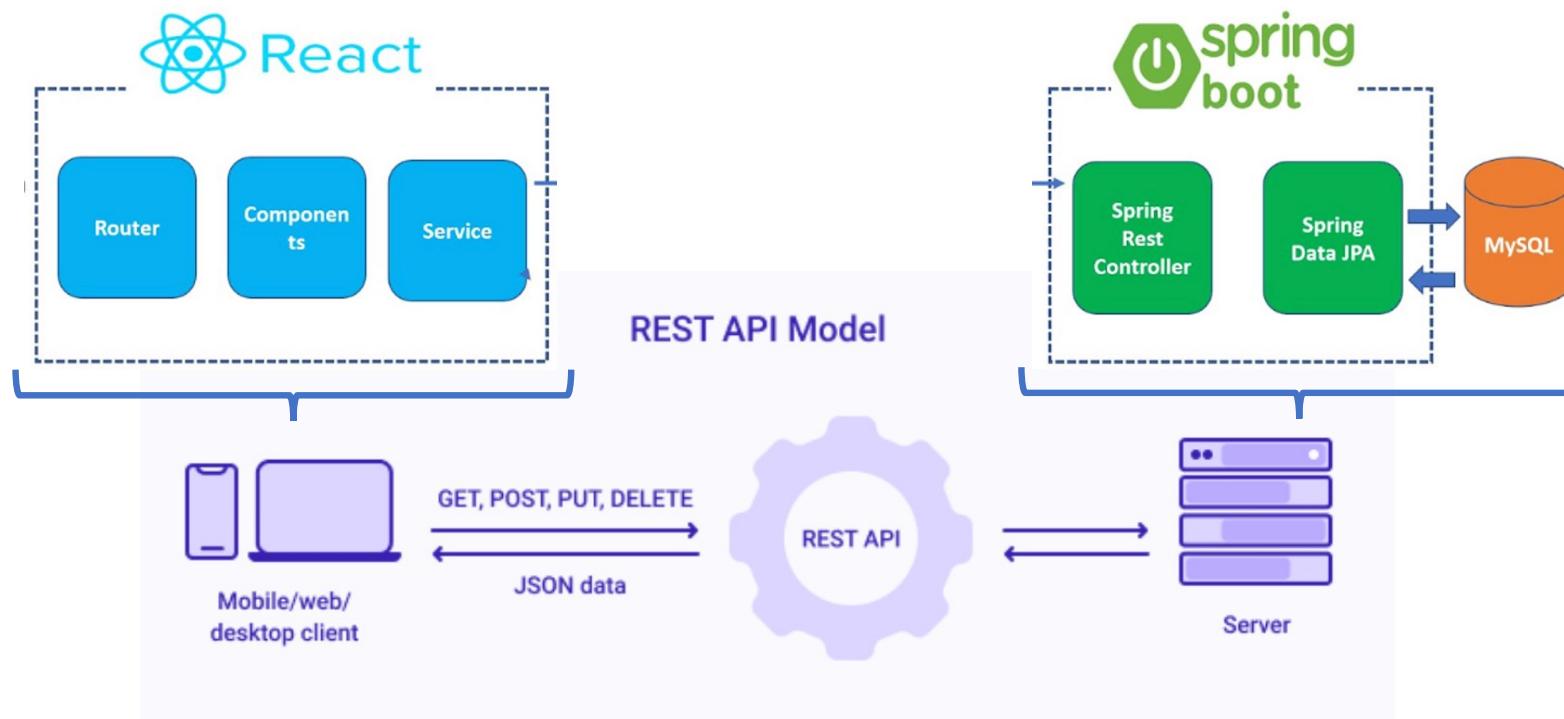
REST API

- **Definição**

- REST é um formato de WebServices que simplifica o uso de recursos computacionais, tais como capacidade de processamento dos dispositivos, bem como o consumo de banda de dados nas redes de comunicações, algo imperativo com a “**explosão**” de uso dos dispositivos móveis.
- Todo REST é um Web Service, mas nem todo Web Service é um REST.
- Com o uso da Arquitetura REST, há uma total desvinculação do **back-end** com o **front-end**.

REST API

- Front-End & Back-End

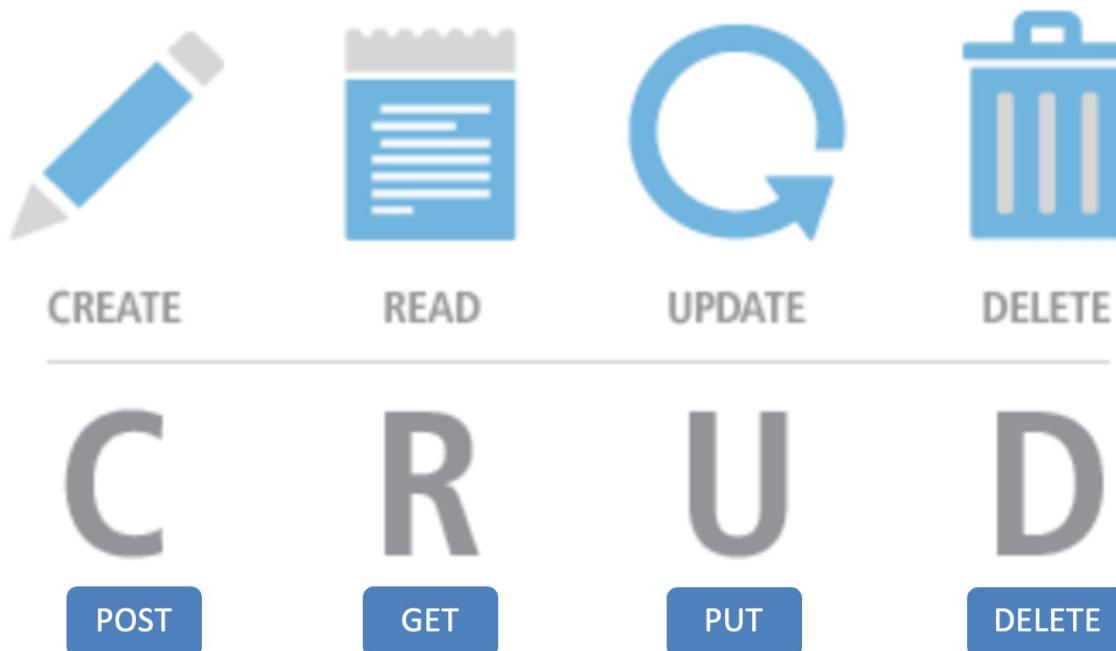


REST API

- **Arquitetura**
 - REST é uma Arquitetura que define um conjunto de boas práticas para a especificação e construção de APIs. Por isso, essas APIs devem ser desenvolvidas dentro das melhores práticas.
 - Utiliza o protocolo **HTTP** para a troca de mensagens (requisição/resposta) e o **JSON** para o envio de dados.
 - Pode ser desenvolvida em qualquer tecnologia (Java, JS, Phyton, PHP, C#, Ruby, etc.)
 - Utiliza verbos HTTP para a realização de operações CRUD.

REST API

- Operações CRUD



REST API

- **Principais Princípios**
 - **(1) Cliente-Servidor**
 - A separação entre cliente e servidor permite que cada um evolua independentemente, com baixo acoplamento entre eles. O cliente não precisa saber como as informações são processadas no servidor e o servidor não precisa saber como o cliente apresenta os dados.
 - **(2) Stateless**
 - Cada solicitação enviada pelo cliente contém todas as informações necessárias para executar a operação solicitada, sem depender de solicitações anteriores. Isso permite que o servidor seja escalável, já que não precisa manter o estado da sessão do cliente.

REST API

- **Principais Princípios**
 - **(3) Cacheable**
 - As respostas das solicitações podem ser armazenadas em cache para melhorar o desempenho e a escalabilidade. Isso permite que os clientes reutilizem as respostas armazenadas em cache, evitando solicitações desnecessárias ao servidor.
 - **(4) Uniform Interface**
 - O cliente e o servidor usam uma interface uniforme para se comunicar, o que simplifica a integração entre diferentes sistemas e aplicativos.

REST API

- **(4) Uniform Interface**
 - A interface uniforme é composta pelos seguintes elementos:
 - **Identificação de recursos:** Cada recurso é identificado por um URI exclusivo.
 - **Manipulação de recursos através de representações:** As solicitações e respostas são feitas através de representações de recursos, como JSON.
 - **Mensagens auto descritivas:** As mensagens de solicitação e resposta contêm informações suficientes para que o receptor entenda o significado da mensagem.
 - **HATEOAS (*Hypermedia as the Engine of Application State*):** As respostas devem conter links que permitam a navegação pelo serviço. Isso permite que o cliente descubra e acesse recursos relacionados sem conhecimento prévio do serviço.

REST API

- **Principais Princípios**
 - (5) **Sistema em camadas:** O sistema pode ser composto de várias camadas, onde cada camada só precisa se comunicar com as camadas adjacentes. Isso permite que o sistema seja escalável e flexível.

VISÃO GERAL DO SPRING FRAMEWORK



SPRING FRAMEWORK

- **O Que é o SPRING FRAMEWORK?**
 - O SPRING FRAMEWORK foi criado em 2002 com o objetivo de facilitar a criação de aplicações JAVA, sendo baseado nos conceitos de Inversão de Controle (IoC) e Injeção de Dependências (ID).
 - Naquela época, a antiga especificação J2EE era muito jovem e complexa, com a necessidade da adoção das tecnologias Java Servlet, JSP e EJB.
- **Principais Recursos**
 - **Bibliotecas flexíveis** – O SPRING fornece bibliotecas de terceiros que ajudam na construção de aplicações baseadas **na nuvem**.



SPRING FRAMEWORK

- **Principais Recursos**
 - **Container IoC** - As propriedades Inversão de Controle (IoC) e Injeção de Dependências (ID) melhoram as suas funcionalidades, garantindo segurança e rapidez de desenvolvimento.
 - **Spring Boot** – Ajuda os desenvolvedores a construir aplicações Web e orientadas a microsserviços muito rapidamente.



SPRING FRAMEWORK

- **Principais Módulos**



Spring Boot

Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.



Spring Framework

Provides core support for dependency injection, transaction management, web apps, data access, messaging, and more.



Spring Cloud Data Flow

Provides an orchestration service for composable data microservice applications on modern runtimes.



Spring Security

Protects your application with comprehensive and extensible authentication and authorization support.



Spring Data

Provides a consistent approach to data access – relational, non-relational, map-reduce, and beyond.



Spring Cloud

Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.



Spring Authorization Server

Provides a secure, light-weight, and customizable foundation for building OpenID Connect 1.0 Identity Providers and OAuth2 Authorization Server products.



Spring for GraphQL

Spring for GraphQL provides support for Spring applications built on GraphQL Java.



SPRING FRAMEWORK

- **Principais Módulos**



Spring Session

Provides an API and implementations for managing a user's session information.



Spring Integration

Supports the well-known Enterprise Integration Patterns through lightweight messaging and declarative adapters.



Spring Batch

Simplifies and optimizes the work of processing high-volume batch operations.



Spring AMQP

Applies core Spring concepts to the development of AMQP-based messaging solutions.



Spring HATEOAS

Simplifies creating REST representations that follow the HATEOAS principle.



Spring REST Docs

Lets you document RESTful services by combining hand-written documentation with auto-generated snippets produced with Spring MVC Test or REST Assured.



Spring Flo

Provides a JavaScript library that offers a basic embeddable HTML5 visual builder for pipelines and simple graphs.



Spring for Apache Kafka

Provides Familiar Spring Abstractions for Apache Kafka.



SPRING FRAMEWORK

- **Principais Módulos**



Spring LDAP

Simplifies the development of applications that use LDAP by using Spring's familiar template-based approach.



Spring Shell

Makes writing and testing RESTful applications easier with CLI-based resource discovery and interaction.



Spring Web Services

Facilitates the development of contract-first SOAP web services.



Spring StateMachine

Provides a framework for application developers to use state machine concepts with Spring applications.



Spring Web Flow

Supports building web applications that feature controlled navigation, such as checking in for a flight or applying for a loan.

VISÃO GERAL DO SPRING BOOT



SPRING BOOT

- **VISÃO GERAL**

- O Java Spring Boot (Spring Boot) é uma ferramenta que torna **mais rápido e fácil o desenvolvimento de aplicações Web e de microsserviços** com o Spring Framework.
- São três os seus principais recursos:
 - (1) Autoconfiguração
 - (2) Abordagem Opinativa
 - (3) Aplicações Autônomas



SPRING BOOT

- **(1) AUTOCONFIGURAÇÃO**

- As Aplicações são inicializadas com dependências pré-definidas do Spring Framework, sem necessidade de configuração manual.
- Por exemplo, para trabalhar com o JPA, basta a inclusão `spring-boot-starter-data-jpa`
- Neste caso, como existe a dependência `spring-boot-starter-data-jpa` no projeto e também tem uma classe anotada com `@Entity`, o Spring Boot vai assumir que será usado o JPA para persistir os dados e irá configurar automaticamente um `EntityManagerFactory`, um `DataSource`, uma `TransactionManager`, entre outras classes relacionadas.



SPRING BOOT

- (1) AUTOCONFIGURAÇÃO

- Vantagens
 - Economia de Tempo
 - Configuração da Aplicação baseada em Bibliotecas
 - Dependências do Maven configuradas automaticamente



- **DEFINIÇÃO**

- O **Maven** é uma ferramenta de gerenciamento de projetos.
- Utilizada para gerenciar projetos que são desenvolvidos usando linguagens JVM como Java, Scala, Groovy etc.
- As principais tarefas de uma ferramenta de gerenciamento de projetos incluem:
 - Construir o Código Fonte; Testar de código-fonte;
 - Empacotar o código-fonte (ZIP, JAR, WAR ou EAR);
 - Lidar com o controle de versão e releases dos artefatos
 - Gerar JavaDocs a partir do código-fonte
 - Gerenciar Dependências do Projeto





SPRING BOOT

- **(2) ABORDAGEM OPINATIVA**

- É possível definir as necessidades do projeto durante o processo de inicialização, durante o qual é possível escolher as várias dependências chamadas de **Spring Starters**.
- O Spring Boot possui mais de **50 Spring Starters**, além de muitos outros starters de terceiros também disponíveis.
- Todos os **Spring Starters** utilizam a seguinte anotação:
 - **spring-boot-starter-XYZ**, onde **XYZ** é o tipo de aplicação que se deseja construir.
 - **Exemplo:** **spring-boot-starter-web** utilizado para construir aplicações RESTful com Spring MVC e Tomcat.



SPRING STARTERS

`spring-boot-starter-web`

`spring-boot-starter-test`

`spring-boot-starter-data-jpa`

`spring-boot-starter-thymeleaf`



SPRING STARTERS

spring-boot-starter-test

```
<artifactId>  
    spring-boot-starter-test  
</artifactId>
```

- ◀ JUnit
- ◀ Mockito
- ◀ Hamcrest
- ◀ Spring core
- ◀ Spring test

spring-boot-starter-data-jpa

```
<artifactId>  
    spring-boot-starter-data-jpa  
</artifactId>
```

- ◀ JDBC
- ◀ Entity manager
- ◀ Transaction API
- ◀ Spring DATA JPA
- ◀ Aspects



SPRING STARTERS

spring-boot-starter-web

```
<artifactId>  
    spring-boot-starter-web  
</artifactId>
```

◀ Spring MVC

◀ REST

◀ Tomcat

◀ Jackson



SPRING BOOT

- **(3) APLICAÇÕES AUTÔNOMAS**

- O Spring Boot ajuda os desenvolvedores a criar aplicações autônomas que são executadas por conta própria, sem depender de um servidor Web externo. Como resultado, é possível iniciar a aplicação em qualquer plataforma simplesmente pressionando o comando Executar.

```
@SpringBootApplication
public class ServidorpublicoBdWebApplication {

    public static void main(String[] args) {
        SpringApplication.run(ServidorpublicoBdWebApplication.class, args);
    }
}
```

- Basta acessar o site **Spring Boot Initializr** e preencher os campos apresentados. Os starters (dependências) também podem ser inseridos de forma muito simples.



SPRING INITIALIZR

<https://start.spring.io/>



Project

- Gradle - Groovy Gradle - Kotlin
 Maven

Language

- Java Kotlin Groovy

Spring Boot

- 3.4.0 (SNAPSHOT) 3.4.0 (M3) 3.3.5 (SNAPSHOT) 3.3.4
 3.2.11 (SNAPSHOT) 3.2.10

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	<input checked="" type="radio"/> Jar <input type="radio"/> War
Java	<input type="radio"/> 23 <input type="radio"/> 21 <input checked="" type="radio"/> 17

Dependencies

[ADD DEPENDENCIES... ⌘ + B](#)

No dependency selected

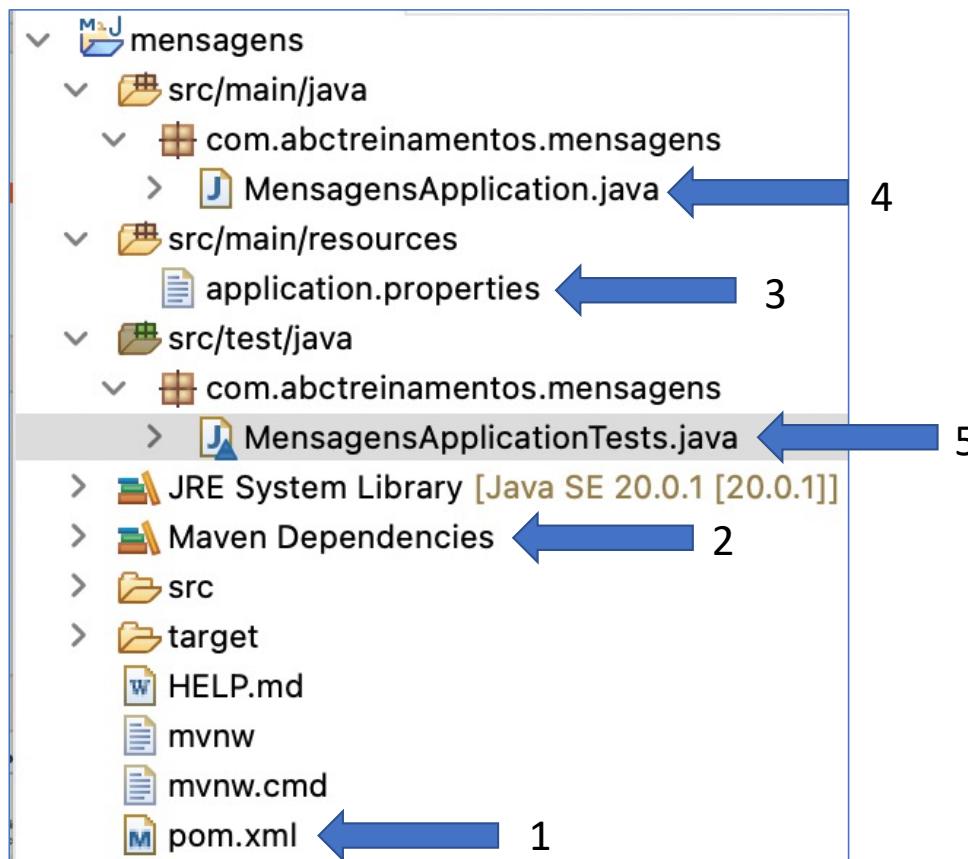
[GENERATE ⌘ + ↩](#)

[EXPLORE CTRL + SPACE](#)

[SHARE...](#)

ESTRUTURA DE UM PROJETO SPRING BOOT

ESTRUTURA DE UM PROJETO SPRING BOOT



(1) ARQUIVO POM.XML

- O **Maven** é uma ferramenta desenvolvida pela Apache para gerenciar projetos Java, organizando as suas dependências e automatizando os seus builds.
- O arquivo **pom.xml** (*Project Object Model*) define todas as configurações do Maven.
- Vamos dividir o arquivo **pom.xml** em 05 partes, para facilitar o seu entendimento:
 - **1ª. Parte:** Define todas as configurações básicas e dependências que serão herdadas em cada projeto Spring Boot

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.0.5</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
```

(1) ARQUIVO POM.XML

- 2ª. Parte: Identificação do Projeto

```
<groupId>com.abctreinamentos</groupId>
<artifactId>mensagensWeb</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>mensagensWeb</name>
<description>Envio de mensagens – Spring Boot</description>
```

- 3ª. Parte: Propriedades do Projeto

```
<properties>
    <java.version>20</java.version>
</properties>
```

(1) ARQUIVO POM.XML

- 4ª. Parte: Dependências do Projeto

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

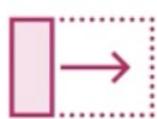
- 5ª. Parte: Plugins necessários para a compilação do Projeto

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

(2) SPRING BOOT MAVEN PLUG-IN

- O **Spring Boot Maven Plugin** é um plugin Maven que fornece suporte para compilar e empacotar aplicativos Spring Boot em um único arquivo executável. Ele automatiza muitas tarefas comuns, como a criação de um arquivo JAR executável, a adição de dependências necessárias ao arquivo JAR e a execução do aplicativo.
- O plugin fornece vários objetivos do Maven que permitem realizar tarefas relacionadas ao Spring Boot, como a execução do aplicativo Spring Boot, a execução de testes de integração do Spring Boot e a execução de um servidor Spring Boot embutido.

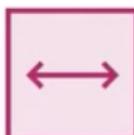
(2) SPRING BOOT MAVEN PLUG-IN



- Faz o empacotamento dos arquivos .JAR e .WAR para serem executáveis



- Roda Aplicações Spring Boot



- Fornece os *Spring Boot Starters* - que são um conjunto de dependências pré-configuradas para casos de uso comuns, como aplicativos da Web, acesso ao banco de dados, segurança, etc.



- Gerencia o Ciclo de Vida de uma Aplicação Spring Boot

(2) DEPENDÊNCIAS MAVEN

✓ Maven Dependencies

- > spring-boot-starter-web-3.0.5.jar - /Users/verena...
- > spring-boot-starter-3.0.5.jar - /Users/verena...
- > spring-boot-3.0.5.jar - /Users/verenasampaio/...
- > spring-boot-autoconfigure-3.0.5.jar - /Users/...
- > spring-boot-starter-logging-3.0.5.jar - /Users/...
- > logback-classic-1.4.6.jar - /Users/verena...
- > logback-core-1.4.6.jar - /Users/verena...
- > log4j-to-slf4j-2.19.0.jar - /Users/verena...
- > log4j-api-2.19.0.jar - /Users/verena...
- > jul-to-slf4j-2.0.7.jar - /Users/verena...
- > jakarta.annotation-api-2.1.1.jar - /Users/verena...
- > snakeyaml-1.33.jar - /Users/verena...
- > spring-boot-starter-json-3.0.5.jar - /Users/verena...
- > jackson-databind-2.14.2.jar - /Users/verena...
- > jackson-annotations-2.14.2.jar - /Users/verena...
- > jackson-core-2.14.2.jar - /Users/verena...
- > jackson-datatype-jdk8-2.14.2.jar - /Users/verena...
- > jackson-datatype-jsr310-2.14.2.jar - /Users/verena...
- > jackson-module-parameter-names-2.14.2.jar - /Users/verena...

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

- > spring-boot-starter-tomcat-3.0.5.jar - /Users/verena...
- > tomcat-embed-core-10.1.7.jar - /Users/verena...
- > tomcat-embed-el-10.1.7.jar - /Users/verena...
- > tomcat-embed-websocket-10.1.7.jar - /Users/verena...
- > spring-web-6.0.7.jar - /Users/verena...
- > spring-beans-6.0.7.jar - /Users/verena...
- > micrometer-observation-1.10.5.jar - /Users/verena...
- > micrometer-commons-1.10.5.jar - /Users/verena...
- > spring-webmvc-6.0.7.jar - /Users/verena...
- > spring-aop-6.0.7.jar - /Users/verena...
- > spring-context-6.0.7.jar - /Users/verena...
- > spring-expression-6.0.7.jar - /Users/verena...
- > spring-boot-starter-test-3.0.5.jar - /Users/verena...
- > spring-boot-test-3.0.5.jar - /Users/verena...
- > spring-boot-test-autoconfigure-3.0.5.jar - /Users/verena...
- > json-path-2.7.0.jar - /Users/verena...
- > json-smart-2.4.10.jar - /Users/verena...
- > accessors-smart-2.4.9.jar - /Users/verena...

(3) ARQUIVO APPLICATION.PROPERTIES

- O arquivo **application.properties** é um arquivo de configuração utilizado pelo Spring Boot para configurar diferentes aspectos da aplicação, como conexão de banco de dados, configurações de servidor web, propriedades do ambiente, entre outros.
- Ao executar uma aplicação Spring Boot, o arquivo **application.properties** é lido automaticamente pelo framework e as configurações definidas nele são aplicadas na aplicação.

.A.1. Core Properties

- .A.2. Cache Properties
- .A.3. Mail Properties
- .A.4. JSON Properties
- .A.5. Data Properties
- .A.6. Transaction Properties
- .A.7. Data Migration Properties
- .A.8. Integration Properties

.A.9. Web Properties

- .A.10. Templating Properties
- .A.11. Server Properties
- .A.12. Security Properties
- .A.13. RSocket Properties
- .A.14. Actuator Properties
- .A.15. Devtools Properties
- .A.16. Testing Properties

(4) CLASSE MENSAGENSAPPLICATION.JAVA

- É a Classe Principal do 1º Projeto Spring Boot.

```
@SpringBootApplication
public class MensagensApplication {

    public static void main(String[] args) {
        System.setProperty("java.awt.headless", "false");
        SpringApplication.run(MensagensApplication.class, args);
        JOptionPane.showMessageDialog(null, "Primeiro Projeto Spring Boot");
        System.out.println("Primeiro Projeto Spring Boot");
        create(); read(); update(); delete();
    }

    public static void create() {
        System.out.println("Criação de um Registro");
    }

    public static void read() {
        System.out.println("Leitura de um Registro");
    }

    public static void update() {
        System.out.println("Atualização de um Registro");
    }

    public static void delete() {
        System.out.println("Exclusão de um Registro");
    }
}
```

(5) CLASSE MENSAGENSAPPLICATIONTESTS.JAVA

- É a Classe Principal de Testes do 1º Projeto Spring Boot.

```
@SpringBootTest
class MensagensApplicationTests {

    @Test
    void contextLoads() {
    }
}
```

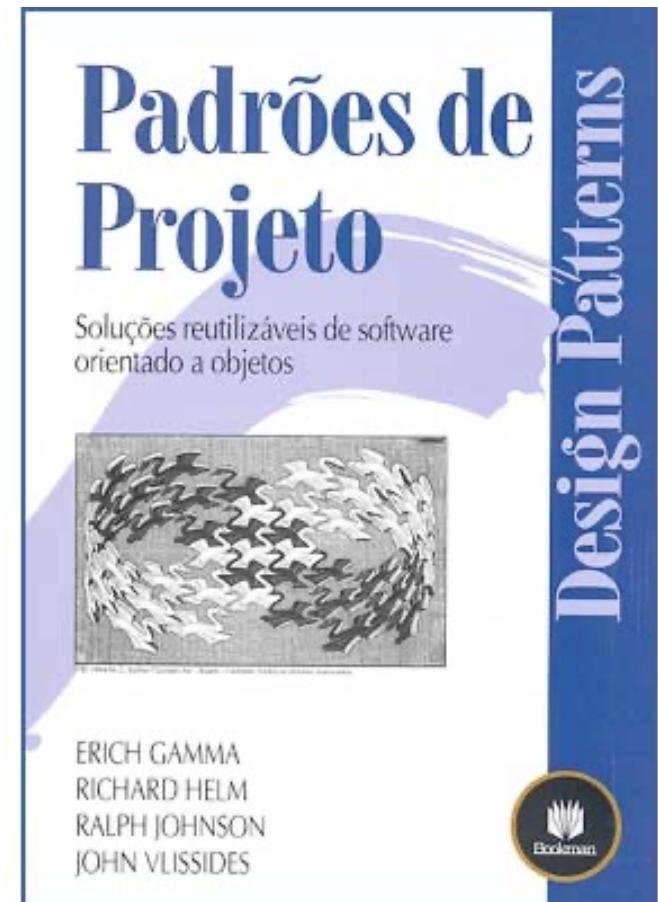
PADRÕES DE PROJETO IOC & DI

PADRÕES DE PROJETO

- **Definição**
 - Padrões de projeto de software, também conhecidos como *Design Patterns*, são soluções comuns e testadas para problemas recorrentes no desenvolvimento de software. Eles são uma forma de capturar a experiência e o conhecimento coletivo de desenvolvedores que enfrentaram e resolveram problemas semelhantes no passado.
 - Os padrões de projeto fornecem uma linguagem comum para desenvolvedores, permitindo que eles comuniquem de forma eficaz soluções e estratégias de projeto. Eles ajudam a evitar reinventar a roda e podem aumentar a eficiência e a qualidade do desenvolvimento de software.

PADRÕES DE PROJETO

- **Definição**
 - Existem vários tipos de padrões de projeto, como padrões de criação, padrões estruturais e padrões comportamentais. Cada tipo aborda um conjunto diferente de problemas no desenvolvimento de software. Alguns exemplos de padrões de projeto famosos incluem Singleton, Factory Method, **MVC**, Observer, Strategy, Adapter, **Injeção de Dependência**, **Inversão de Controle**, entre outros.
- O livro ao lado é a referência no estudo de padrões de projeto na área de engenharia de software.





INVERSÃO DE CONTROLE

- **Definição**

- Inversão de Controle (*Inversion of Control* - IoC) é um princípio de design de software que permite criar sistemas mais flexíveis e com menor acoplamento entre os componentes. A ideia é que, em vez de cada componente do sistema ser responsável por instanciar e gerenciar suas próprias dependências, essa responsabilidade é transferida para um componente externo, chamado de container de IoC.
- O container de IoC é responsável por gerenciar as dependências dos componentes do sistema e, assim, controlar o fluxo de execução da aplicação. Ele provê aos componentes as dependências necessárias e pode substituí-las ou modificar seu comportamento de acordo com a configuração da aplicação.



INVERSÃO DE CONTROLE

- **Definição**

- Dessa forma, os componentes podem ser desenvolvidos de forma mais modular e reutilizável, facilitando a manutenção e evolução do sistema como um todo.
- A inversão de controle é frequentemente associada ao padrão de projeto Injeção de Dependência (*Dependency Injection - DI*), que é uma técnica utilizada para implementar a IoC.
- Com a DI, as dependências de um componente são "injetadas" nele pelo container de IoC, em vez de serem criadas internamente pelo componente. Isso torna o componente mais flexível, pois ele não precisa saber como suas dependências são criadas, apenas como utilizá-las.



INJEÇÃO DE DEPENDÊNCIA

- **Definição**

- É um padrão de projeto usado para **evitar o alto nível de acoplamento de código** dentro de uma aplicação. Sistemas com baixo acoplamento de código são melhores pelos seguintes motivos: aumento na facilidade de manutenção/implementação de novas funcionalidades e também facilita a realização de testes unitários.

```
public class Pedido {  
    public static void main(String[] args) {  
        List<Produto> produto = new ArrayList<>();  
        Produto roupa = new Produto();  
        produto.add(roupa);  
    }  
}
```

- A **Classe Pedido** está **altamente “acoplada”** com a **Classe Produto**. Se houver uma alteração no construtor de Produto, a classe Pedido também deverá ser alterada.



INJEÇÃO DE DEPENDÊNCIA

- A Injeção de Dependência (DI) é uma das formas de se implementar a IoC. A outra forma é utilizar o padrão de projeto *Service Locator*.
- Existem quatro maneiras de implementar a DI:
 - Construtor
 - Setter
 - Interface
 - Anotações



INJEÇÃO DE DEPENDÊNCIA

- **Construtor**

- A dependência é fornecida como parâmetro do construtor da classe. É a forma mais comum de realizar a injeção de dependência.

```
public class Pedido {  
  
    private List<Produto> produto;  
  
    public Pedido (List<Produto> produto)  
    {  
        this.produto = produto;  
    }  
}
```



INJEÇÃO DE DEPENDÊNCIA

- **Construtor**

- E se houver mais de um construtor (sobrecarga), como o Spring saberá qual será o ponto de injeção?

```
public class Pedido {  
  
    private List<Produto> produto;  
  
    // Esse é o ponto de injeção  
    @Autowired  
    public Pedido (List<Produto> produto)  
    {  
        this.produto = produto;  
    }  
  
    public Pedido (String nomeProduto)  
    {  
        // sobrecarga  
    }  
}
```



INJEÇÃO DE DEPENDÊNCIA

- **Método Setter**

- A dependência é definida por meio de um método setter. Essa técnica permite que as dependências sejam alteradas a qualquer momento durante a execução do programa.

```
public class Pedido {  
  
    private List<Produto> produto;  
  
    // Esse é o ponto de injeção  
    @Autowired  
    public void setProduto(List<Produto> produto)  
    {  
        this.produto = produto;  
    }  
}
```



INJEÇÃO DE DEPENDÊNCIA

- **Interface**

- Uma interface é definida para a dependência, e a classe que precisa dessa dependência é projetada para implementar essa interface.

```
public class Pedido {  
    private List<IProduto> produto;  
    public Pedido (List<IProduto> produto)  
    {  
        this.produto = produto;  
    }  
}
```



INJEÇÃO DE DEPENDÊNCIA

- **Anotações**

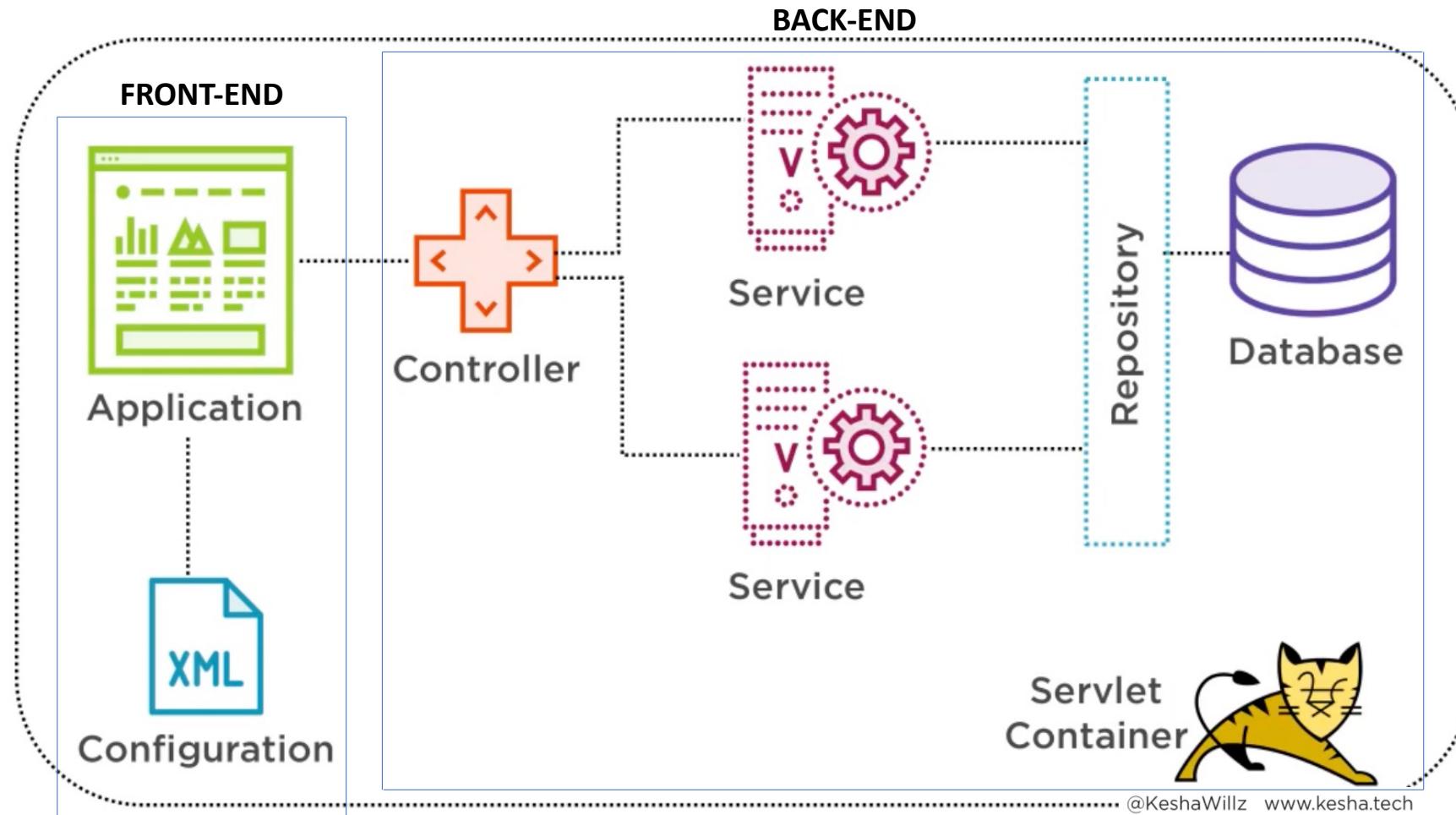
- Anotações são usadas para identificar as dependências que precisam ser injetadas. O container de injeção de dependência examina as anotações e injeta as dependências correspondentes automaticamente.

```
public class Pedido {  
  
    // Esse é o ponto de injeção  
    @Autowired  
    private List<Produto> produto;  
  
    public List<Produto> produtos()  
    {  
        return produto;  
    }  
}
```

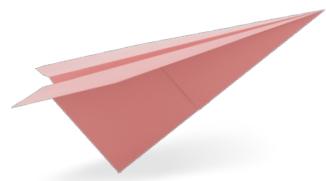
ARQUITETURA SPRING BOOT



ARQUITETURA SPRING BOOT

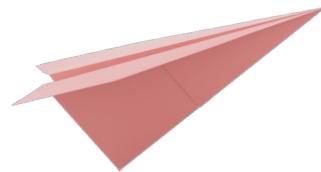


PROJETO PRÁTICO



Projeto Spring Boot – Aplicação Servidor Público/Curso REST API com THYMELEAF

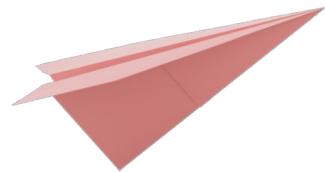




TELA PRINCIPAL

Foto	Nome	Cargo	Órgão	Lotação	Email	Cursos	Ações		
	Isabela Sampaio	Auditor	ANVISA	Brasília	assoftbel@gmail.com	Não Tem Cursos	Detalhar	Alterar	Excluir
	Heila Ghassan	Estagiário	STN	Brasília	heila@gmail.com	Não Tem Cursos	Detalhar	Alterar	Excluir
	Maria Fontenele	Analista	ENAP	Brasília	mariafontenele@enap.br	Cursos Matriculados	Detalhar	Alterar	Excluir
	Caio Santos	Analista Tributário	RFB	Rio de Janeiro	caiosantos@rfb.gov.br	Cursos Matriculados	Detalhar	Alterar	Excluir





TELA DO SERVIDOR

Isabela Sampaio

Informações Funcionais

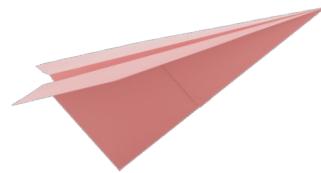
Matrícula: 6	Lotação: Brasília
Órgão: ANVISA	Exercício: SUPEN
Cargo: Auditor	Vínculo: Estatutário
Data de Admissão: 2022-11-21T12:55:40.496Z	

Informações Pessoais

Email: asoftbel@gmail.com	CPF: 00000000000
Telefone: (91)991124552	Data de Nascimento: 2022-08-03T12:55:40.496Z
Celular: (91)981144444	Naturalidade: Belém

[Voltar](#)



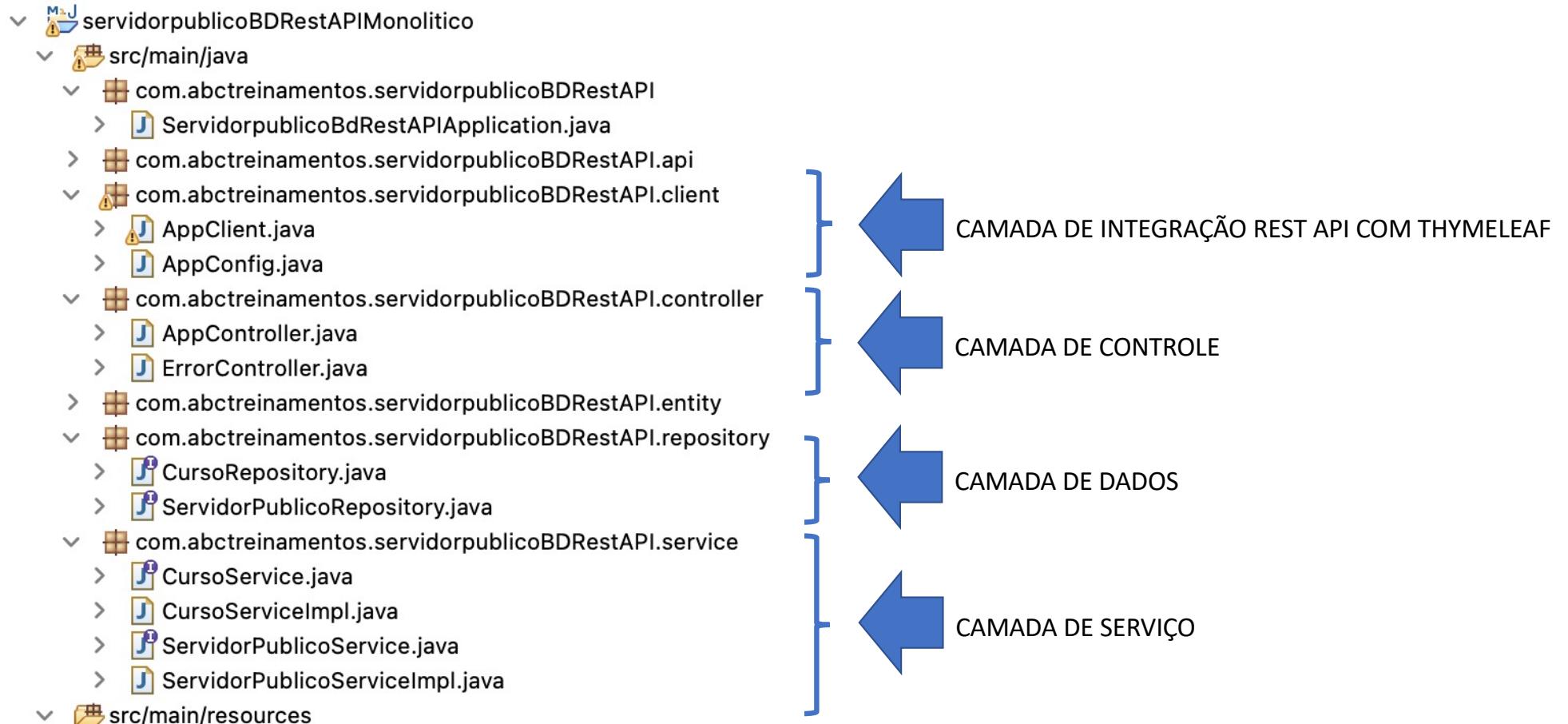


SERVIDORPUBLICO.JSON

```
[  
  {  
    "matricula": 3,  
    "nome": "Maria Fontenele",  
    "foto": "https://abctreinamentos.com.br/imgs/maria.png",  
    "orgao": "ENAP",  
    "vinculo": "Estatutário",  
    "cargo": "Analista",  
    "lotacao": "Brasília",  
    "exercicio": "Departamento de Treinamento",  
    "email": "mariafontenele@enap.br",  
    "telefone": "(61) 3255-6010",  
    "celular": "(61) 99910-5722",  
    "cpf": "123.4567.789-01",  
    "naturalidade": "Recife",  
  }  
]
```

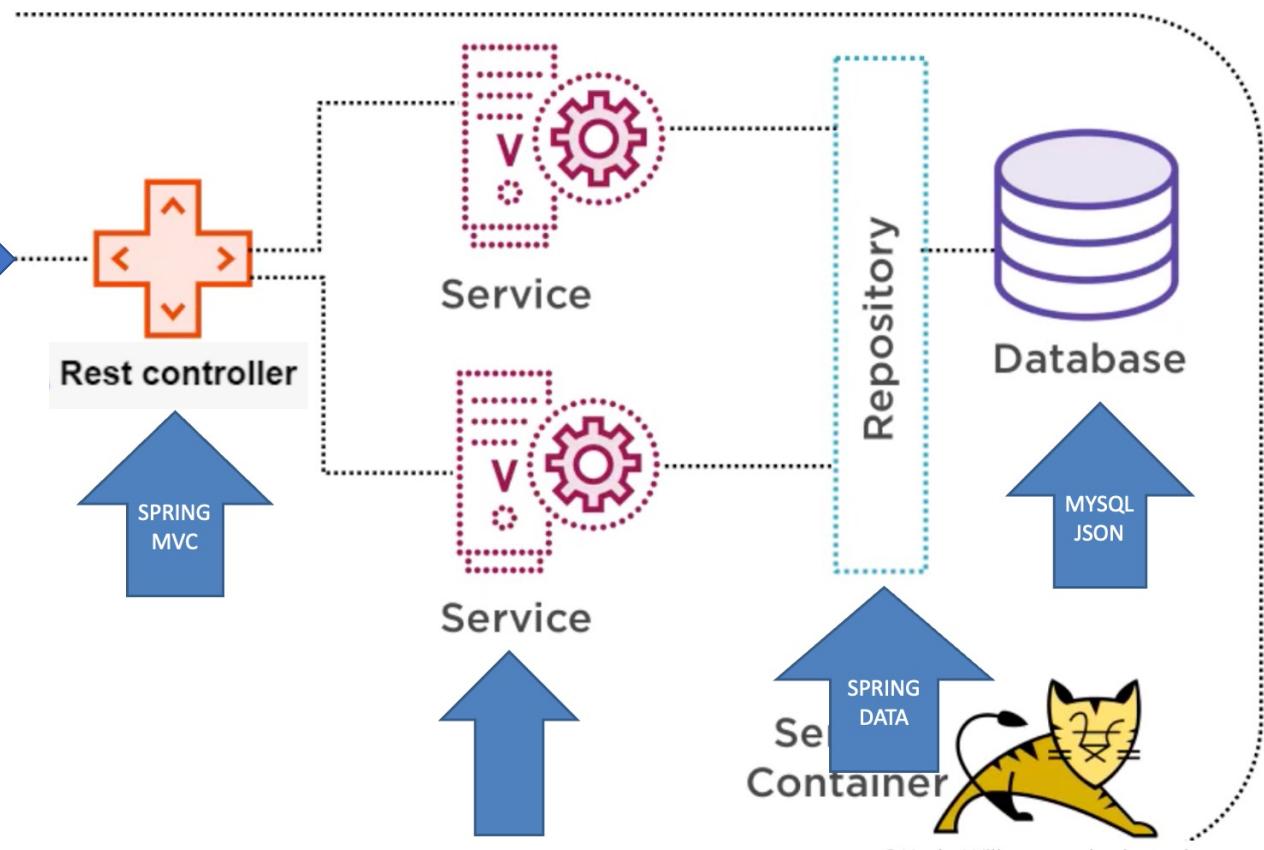
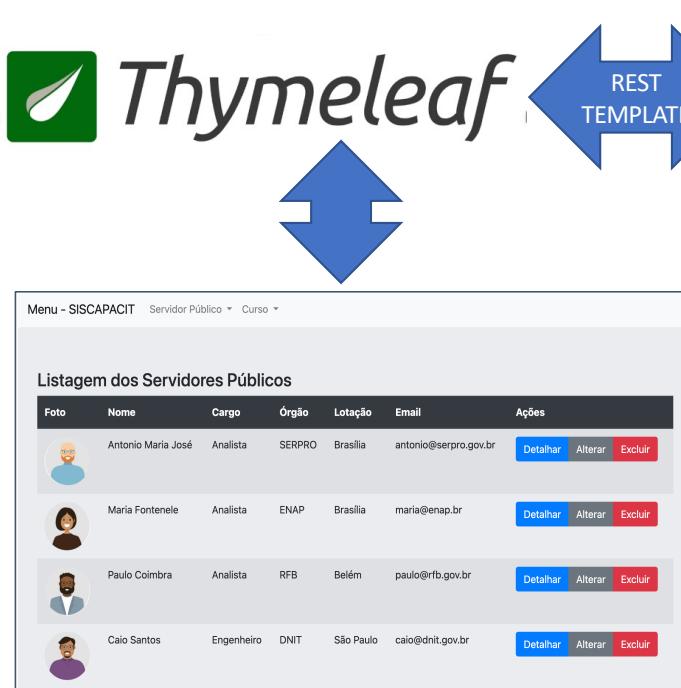


ARQUITETURA MONOLÍTICA



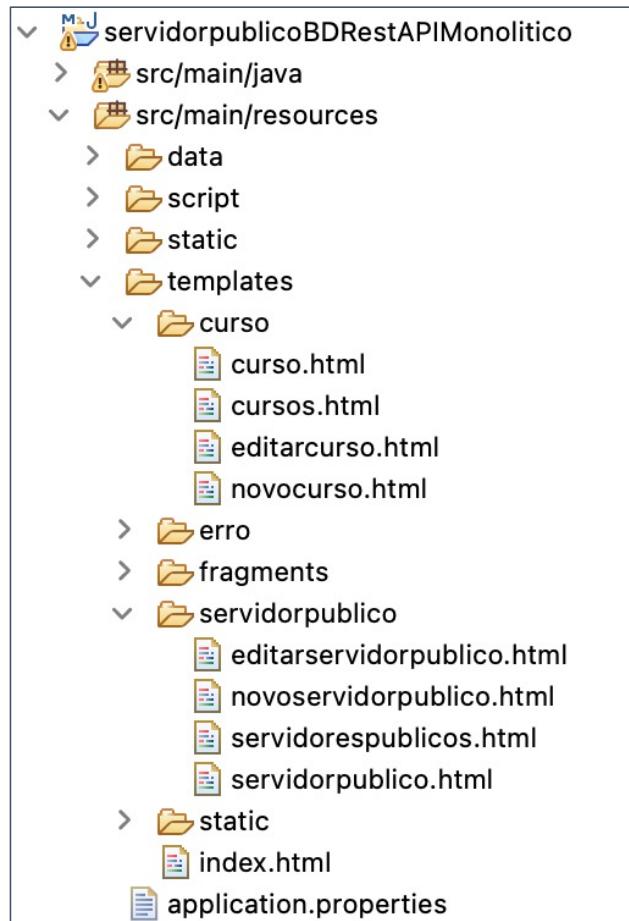


ARQUITETURA THYMELEAF



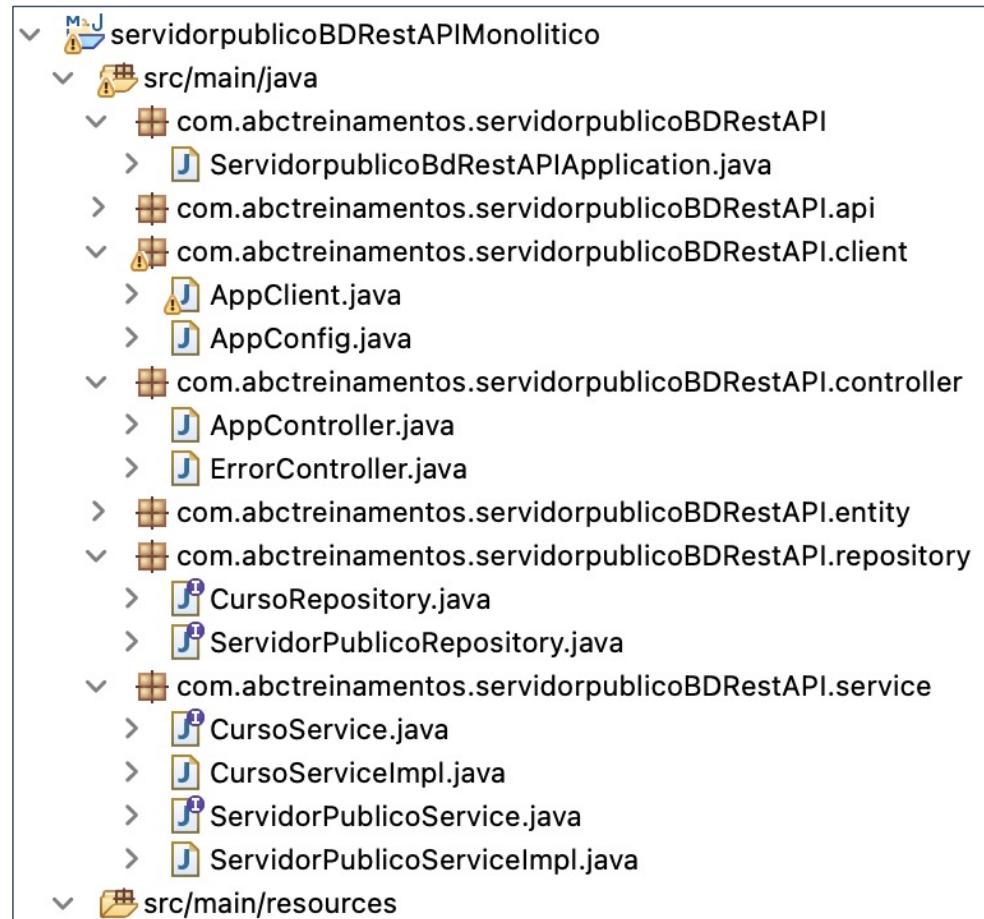


ARQUITETURA DO PROJETO - FRONTEND





ARQUITETURA DO PROJETO - BACKEND



CURSO Aplicações JAVA com SPRING BOOT

- **Principais Diferenciais**

- 01 eBook com 456 páginas de conteúdo atualizado para o Spring Boot 3
- 09 Unidades organizadas de forma a facilitar o aprendizado do aluno
- 77 videoaulas que totalizam muitas e muitas horas de curso
- 12 projetos com milhares de linhas de código



PARCERIA EXCLUSIVA
FORMAÇÃO COMPLETA

Profissão: Desenvolvedor Full Stack Java

Desenvolva habilidades completas em programação, abrangendo todas as etapas do desenvolvimento de softwares e aplicativos, desde a interface até o sistema e bancos de dados. Domine Java para criar soluções complexas de ponta a ponta, impulsionando suas habilidades como desenvolvedor Full Stack. Esta é uma oportunidade exclusiva para colaboradores da Stefanini.

<https://amazoncode.com.br/stefanini/>

