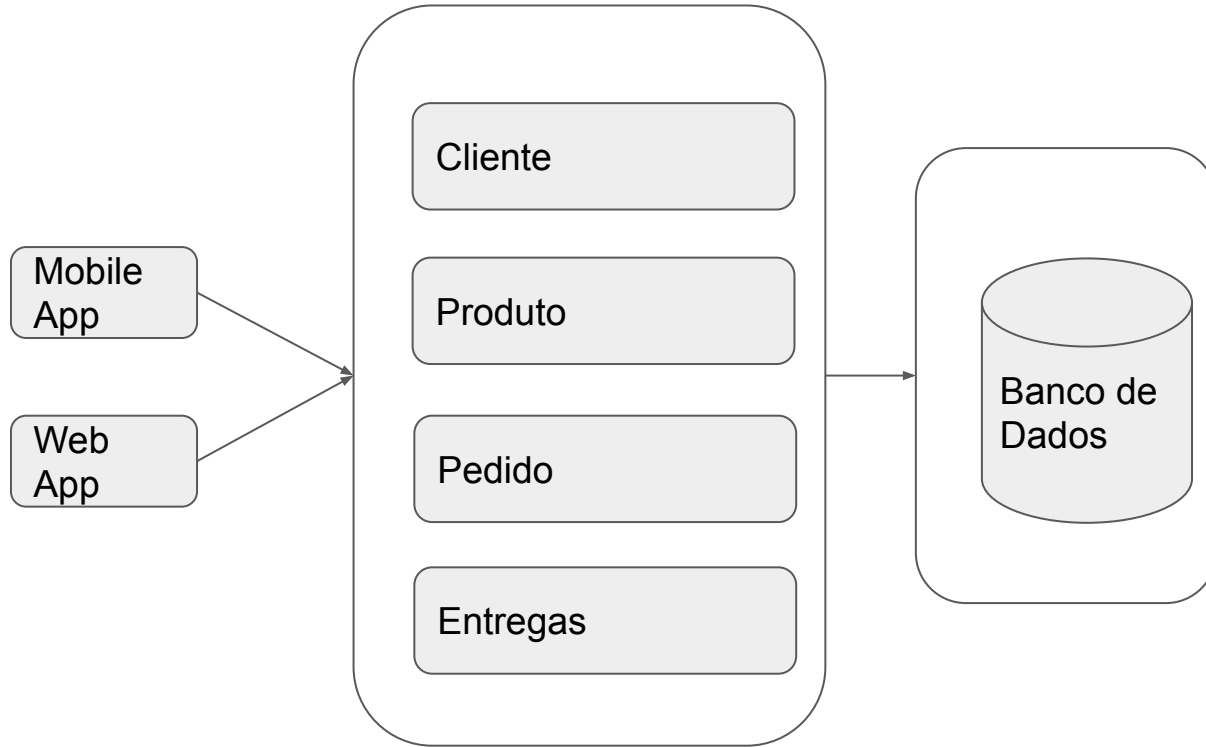


# Microserviços com Spring Boot

Construindo microserviços com Spring Boot e Spring Cloud

# Exemplo de e-commerce **monolítico**



# Vantagens e desafios do monolítico

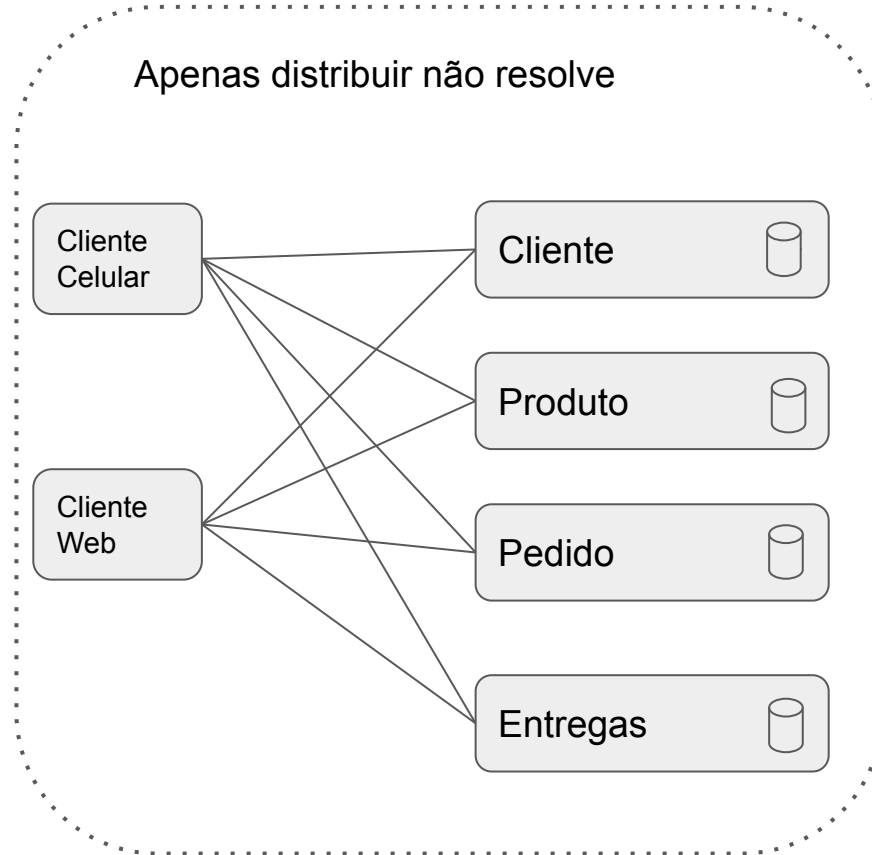
## Vantagens:

- Fácil manutenção
- Maior controle transacional
- Conciso
- Infraestrutura Simplificada
- Recursos em um só lugar
- Baixa integração

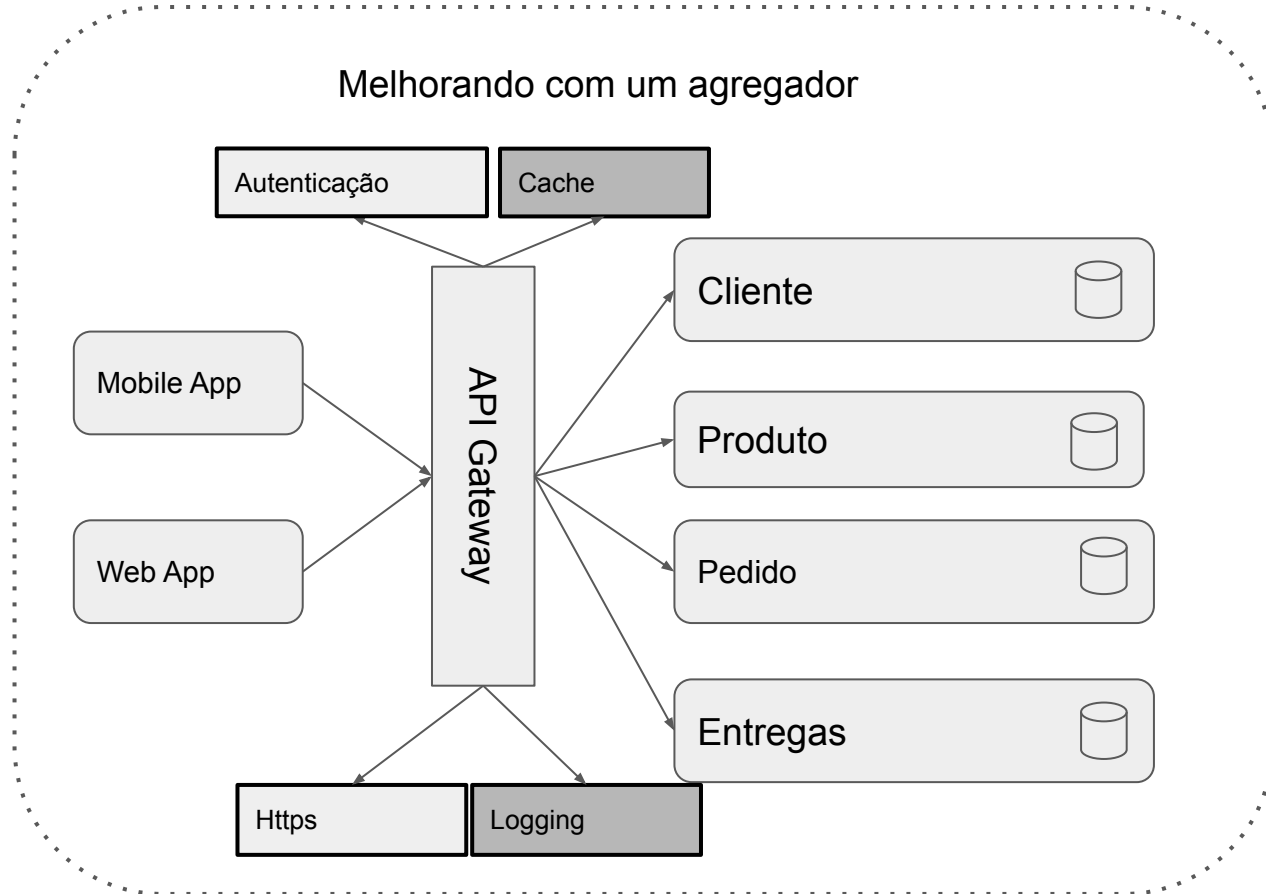
## Desafios:

- Complexidade cresce com o tamanho da aplicação
- Dependência de componentes
- Dificuldade de implantação
- Baixa flexibilidade
- Difícil de escalar
- Difícil de manter
- Curva de aprendizado alta para novos devs
- Difícil **escalar times**

# E-commerce com **Microserviços**



# Exemplo de e-commerce com **microserviços**



# Trade-offs dos Microsserviços

## Vantagens:

- Agilidade
- Resiliência flexível
- Melhor escalabilidade
- Flexibilidade e desacoplamento
- Independência de linguagem
- *Time to Market* mais rápido
- Fácil de depurar
- Facilita a entrega contínua

## Desafios:

- Exige implantação rápida
- Exige melhor monitoramento
- Estrutura mais complexa (distribuída)
- Complexidade de ferramentas
- Dados distribuídos
- Infraestrutura distribuída
- Ferramentas de agregação

# Colocando a mão na massa: criando o projeto pelo starter do Spring

É possível gerar seu projeto inicial direto do portal *starter* do Spring, independente da IDE que esteja utilizando.,

- <https://start.spring.io/>
- À partir do portal basta preencher os detalhes do projeto conforme a figura

The screenshot shows the Spring Boot starter web form with the following configuration:

- Project:**  Gradle - Groovy,  **Maven**
- Language:**  **Java**,  Kotlin,  Groovy
- Spring Boot:**  3.1.1 (SNAPSHOT),  3.1.0,  3.0.8 (SNAPSHOT),  3.0.7,  **2.7.13 (SNAPSHOT)**,  2.7.12
- Project Metadata:**
  - Group:
  - Artifact:
  - Name:
  - Description:
  - Package name:
  - Packaging:  **Jar**,  War
  - Java:  20,  17,  **11**,  8

Red arrows in the image point to the selected options: Maven, Java, 2.7.13 (SNAPSHOT), Jar, and 11.

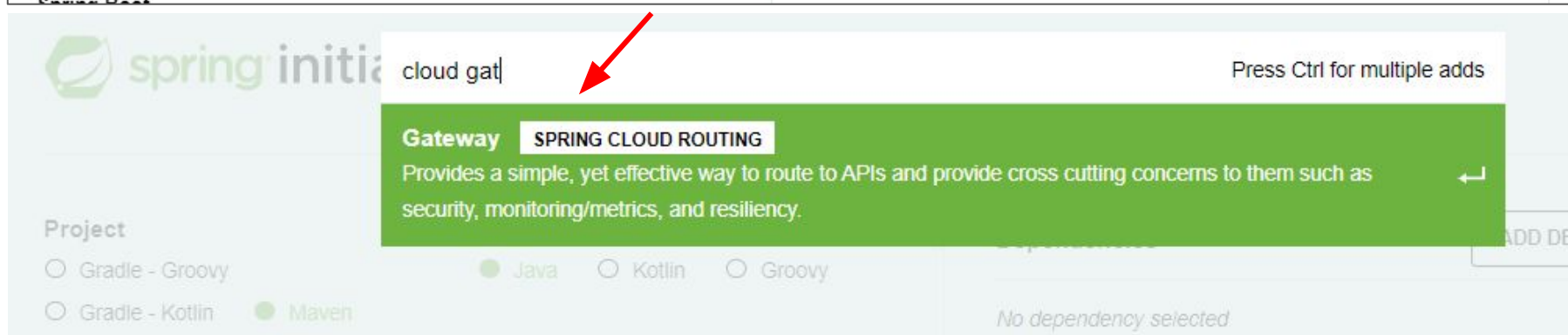
# Spring cloud: adicionando novas dependências

Apesar do nome “Cloud”, a maioria das dependências do Spring Cloud são para facilitar a implementação de microsserviços.

Nas próprias dependências do Spring Starter, você consegue pesquisar pelas funcionalidades. Na figura abaixo, buscamos pelo Api Gateway, que é um Padrão de Microsserviços. Ao criar um projeto novo com esta dependência, ele subirá um servidor implementando esse padrão - você só precisará configurar as rotas.



The screenshot shows the Spring Initializr interface. On the left, under 'Project', 'Gradle - Kotlin' is selected. Under 'Language', 'Java' is selected. On the right, there is a search bar with the text 'cloud gat|' and a red arrow pointing to it. Below the search bar, a dropdown menu is open, showing 'Gateway' with the sub-label 'SPRING CLOUD ROUTING'. The description for 'Gateway' reads: 'Provides a simple, yet effective way to route to APIs and provide cross cutting concerns to them such as security, monitoring/metrics, and resiliency.' To the right of the search bar, there is a button that says 'ADD DEPENDENCIES... CTRL + B' and the text 'No dependency selected'.



This is a close-up of the search results from the previous screenshot. The search bar contains 'cloud gat|'. The dropdown menu is highlighted in green and shows 'Gateway' with the sub-label 'SPRING CLOUD ROUTING'. The description for 'Gateway' reads: 'Provides a simple, yet effective way to route to APIs and provide cross cutting concerns to them such as security, monitoring/metrics, and resiliency.' To the right of the search bar, there is a button that says 'ADD DEPENDENCIES... CTRL + B' and the text 'No dependency selected'.



# Gerando o projeto

Na sessão de dependências você vai adicionando as que precisar para o projeto, como JPA e Spring Data para banco de dados, driver do banco, como H2, Mysql, Postgres e afins.

Depois é só baixar o projeto (botão GENERATE). O projeto compactado será baixado na pasta de downloads. Depois disso, basta descompactá-lo na pasta de sua preferência e abrir na sua IDE favorita.

## Spring Boot

- 3.1.1 (SNAPSHOT)    3.1.0    3.0.8 (SNAPSHOT)    3.0.7  
 2.7.13 (SNAPSHOT)    2.7.12

## Project Metadata

Group

Artifact

Name

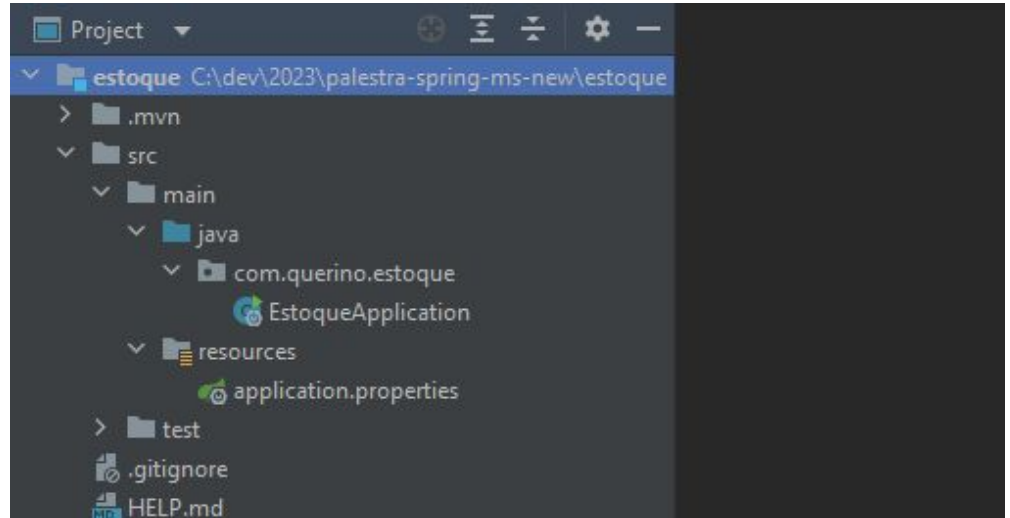
**GENERATE** CTRL + ↵

**EXPLORE** CTRL + SPACE

**SHARE...**

# Importando o projeto

O projeto gerado pelo Spring Starter já vem com as dependências e a estrutura inicial do projeto. A partir daí você já pode importá-lo na sua IDE favorita e começar a programar suas funcionalidades.



# Melhorando a arquitetura:

Existem vários padrões para implementação de microsserviços. A seguir apresento alguns que você pode ir estudando e adicionando à sua implementação para praticar.

O **código fonte com um modelo** pode ser encontrado no GitHub - nele foram adicionados alguns padrões e funcionalidades do Spring Cloud, além do Docker:

<https://github.com/cloud4java/palestra-spring-ms.git>

- Api Gateway
- Service Discovery (Descoberta de Serviços): Eureka, etc
- Load Balance (Distribuição de Carga): Eureka/Spring Load Balance (client side)
- Observability: Trace, Logs/Events, Metrics
- - Tracing: Sleuth + Zipkin /Jaegger, etc
- - Logs: Prometheus / Grafana Loki, etc
- - Metrics: Grafana, etc
- Swagger (Documentação de Api)

# Referências

- **Microserviços** - definições Google  
<https://cloud.google.com/architecture/microservices-architecture-introduction?hl=pt-br>
- **Vantagens e desvantagens:**  
Edu Silveira - 2020: <https://edusilveira.com.br/microservico-vantagens-e-desvantagens>  
Pablo Santos - 2021:  
<https://arphoenix.com.br/quais-as-diferencas-entre-arquitetura-monolitica-e-microservicos-suas-vantagens-e-desvantagens>
- **Livro:** [Jornada Microserviços: do zero ao avançado somando conceitos e práticas](#)
- Docker tutorial: <https://jstobigdata.com/docker/advanced-docker-tutorial/>
- Docker-compose:
- **Observability:**
- (Monolíticos) <https://www.baeldung.com/spring-cloud-sleuth-single-application>
- (Microserviços): <https://www.baeldung.com/tracing-services-with-zipkin>
- <https://medium.com/javarevisited/distributed-tracing-in-microservices-spring-boot-125272b58ad8>
- Start Zipkin: <https://zipkin.io/pages/quickstart>
- Splunk:
- <https://splunk.github.io/docker-splunk/EXAMPLES.html#create-standalone-from-compose>
- <https://medium.com/@reddy.srikant/logging-configuration-in-spring-boot-for-splunk-e6b504658011>
-

**Parabéns por mais este aprendizado na sua carreira**

**Saiba mais seguindo nas redes sociais**

Linkedin: dorivalq

Twitter: @dorivalq

Youtube: cloud4java

