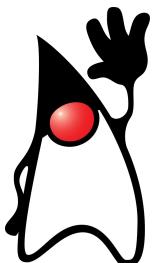


Desenvolvimento com Frameworks e Componentes

Michel Vasconcelos

michel.vasconcelos@gmail.com



Previously on Developing with
Frameworks and Components
@UNI7...

Enterprise Java Beans (EJBs)



Enterprise Java Bean (EJB)

Componentes corporativos que executam no lado servidor, podendo ser disparados local ou remotamente com todo seu ciclo de vida gerenciado por um container EJB e que implementam regras de negócio, colaborando entre si para entregar valor para o usuário final.

Enterprise Java Bean (EJB)

Componentes corporativos que executam no **lado servidor**, Podendo ser disparados **local ou remotamente** com todo seu **ciclo de vida gerenciado por um container EJB** e que implementam **regras de negócio**, colaborando entre si para **entregar valor ao usuário final**.

Porque usar?



Para reflexão



Joe Duffy

@funcOfJoe

25yrs ago: COM (focus on your biz logic)

20yrs ago: Java (focus on your biz logic)

15yrs ago: .NET (focus on your biz logic)

10yrs ago: Dynamic langs (focus on your biz logic)

5yrs ago: Microservices (focus on your biz logic)

0yrs ago: Serverless (focus on your biz logic)

Quando usar?

Aplicação necessita

Distribuição

Escalabilidade

Cont. transacional

Segurança

etc

Tipos

Session Bean: Executa uma ação

Message Driven Bean: Fornece assincronia

Tipos

Session Bean

Message Driven Bean

Entity Bean: Representa uma entidade no BD

Tipos

Session Bean

Message Driven Bean

~~Entity Bean~~

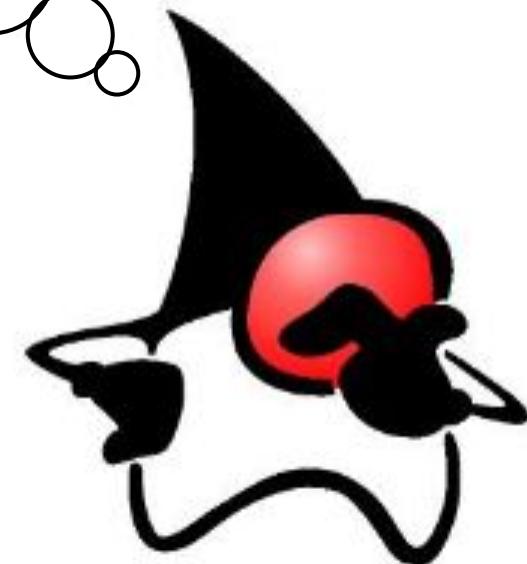
Session Beans



Considerações

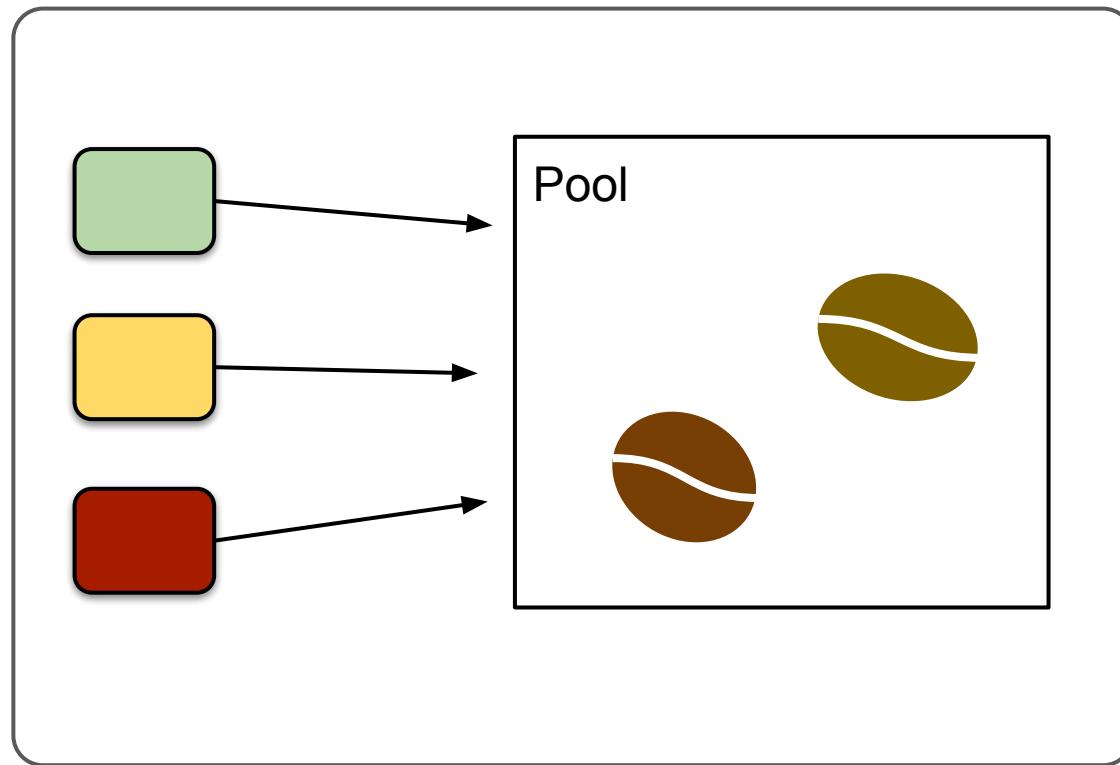
- Encapsula regra de negócio
- Possui três tipos: Stateless, Stateful e Singleton
- Acionamento local ou remoto
- Não são persistentes
- Thread-safe

**Valha... e tem 3 tipos de
SB? Quais as diferenças
mesmo?**

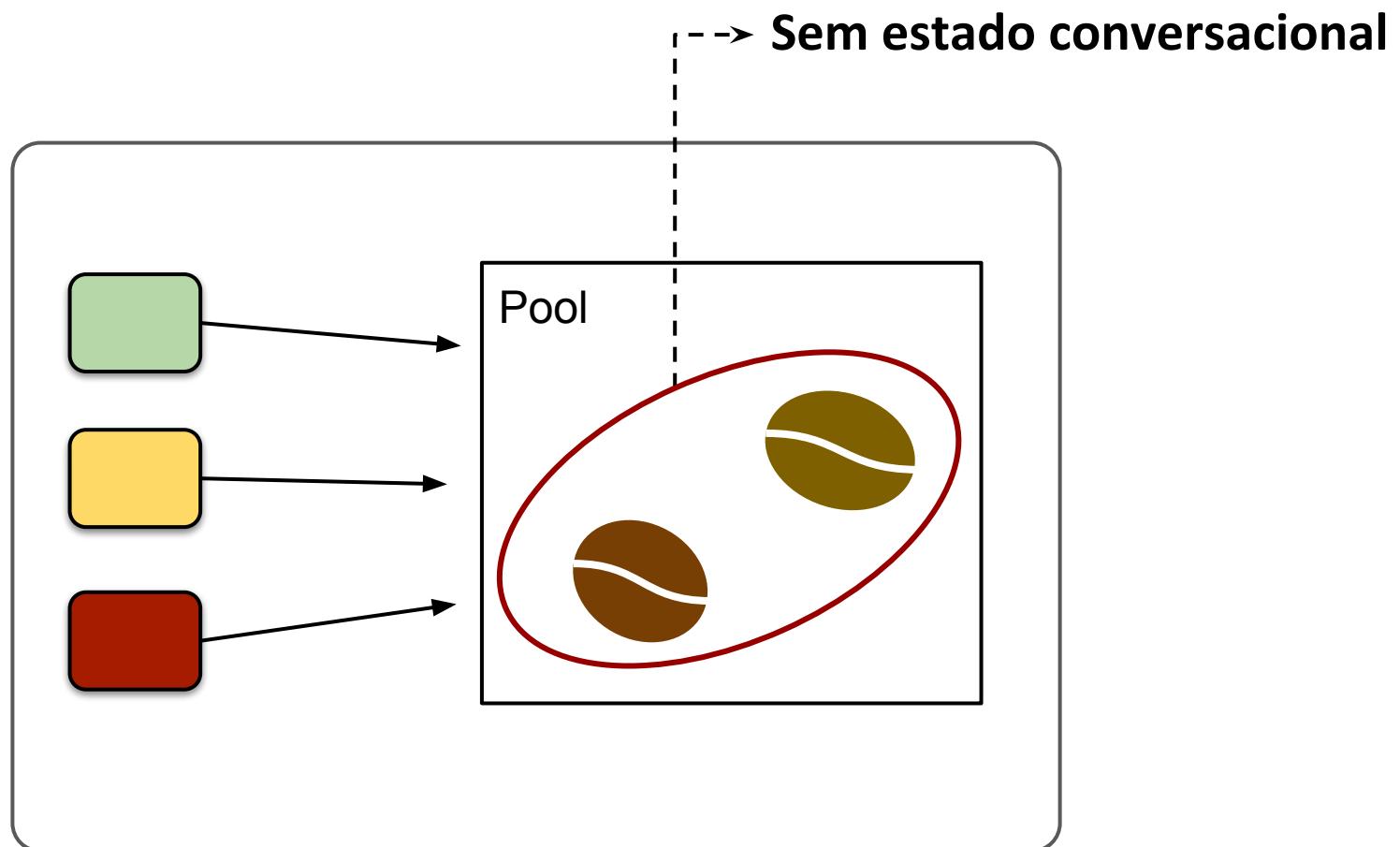




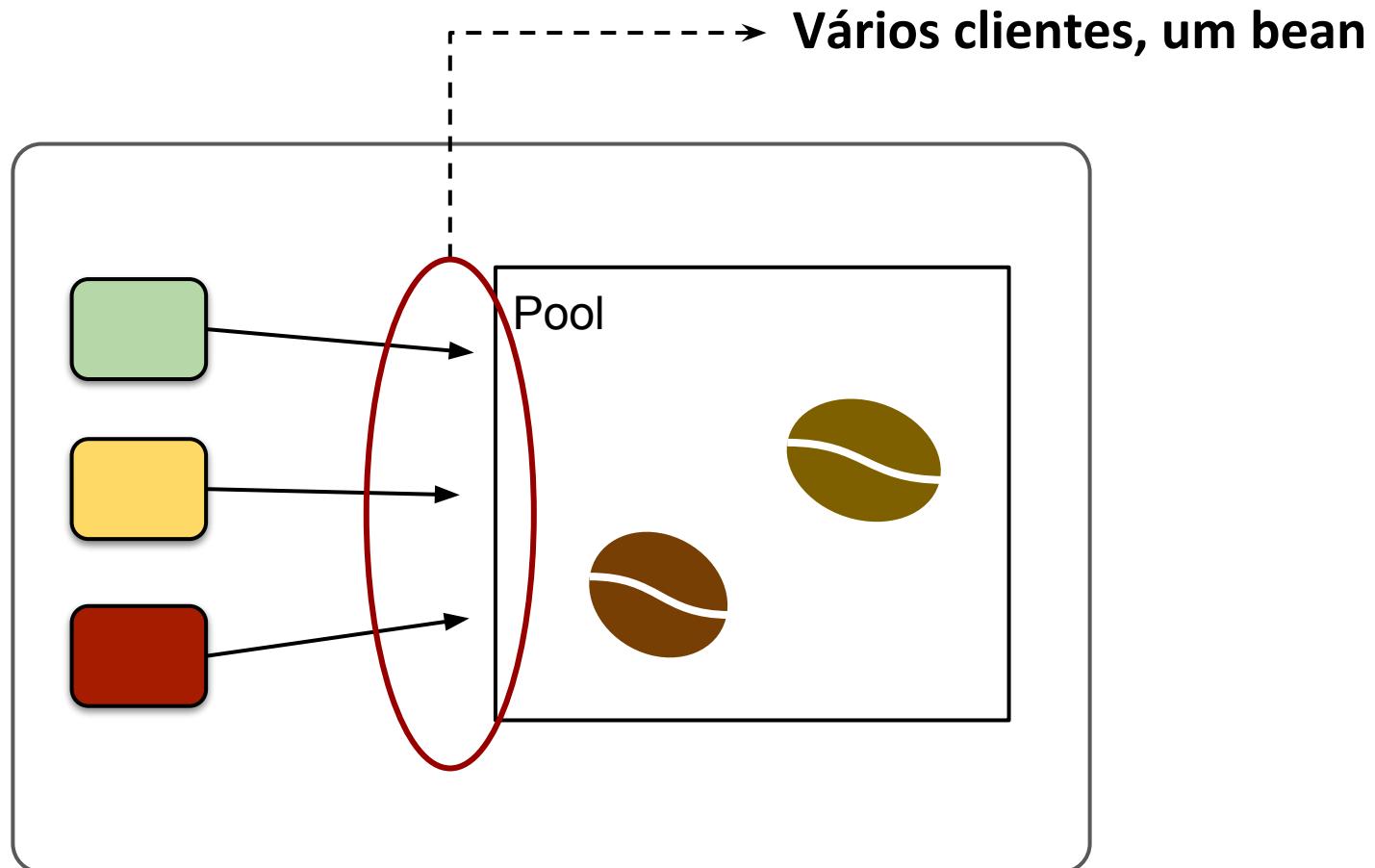
Stateless



Stateless



Stateless



```
@Stateless(name = "MyBean",
           mappedName = "MyMappedName",
           description = "Desc",
           passivationCapable = true)
public class MyBusinessBean implements MyLocalInterface {

    private Map counts = new HashMap();

    public void process(String msg) {
        // Do something
    }

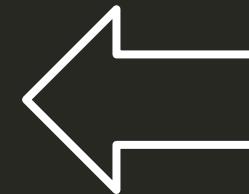
    public Map<String, Integer> counts() {
        // Do something
    }
}
```

```
@Stateless(name = "MyBean",
           mappedName = "MyMappedName",
           description = "Desc",
           passivationCapable = true)
public class MyBusinessBean implements MyLocalInterface {

    private Map counts = new HashMap();

    public void process(String msg) {
        // Do something
    }

    public Map<String, Integer> counts() {
        // Do something
    }
}
```



```
@WebServlet(name = "myServlet", urlPatterns = {"/hello"} )
public class MyServlet extends HttpServlet {
    @Inject private MyResource resource;
    @EJB private MyBusinessBean bean;

    protected void doGet(...) {
        // Process Get
    }

    protected void doPut(...) {
        // Process Put
    }
}
```

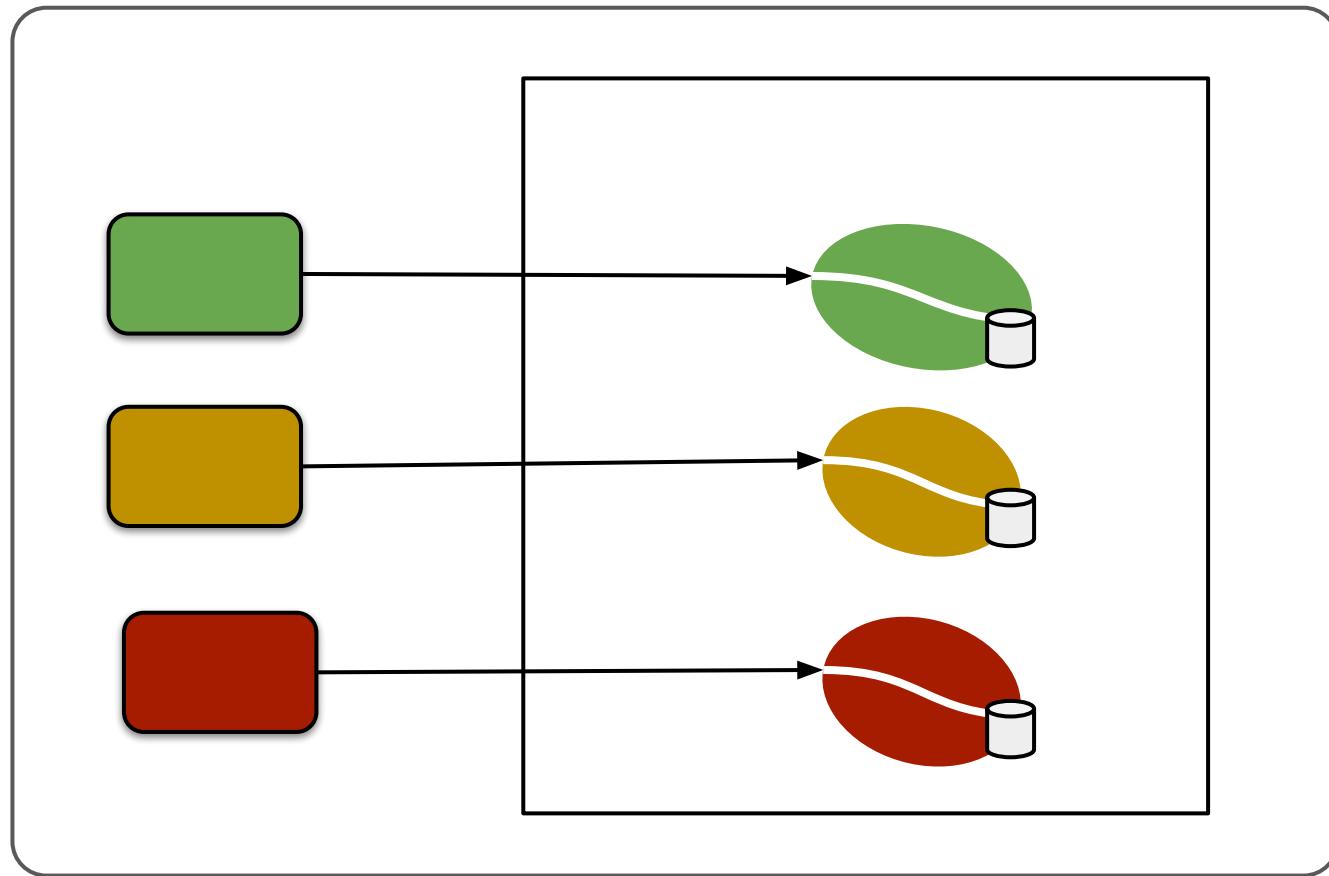
```
@WebServlet(name = "myServlet", urlPatterns = {"/hello"} )  
public class MyServlet extends HttpServlet {  
    @Inject private MyResource resource;  
    @EJB private MyBusinessBean bean;  
  
    protected void doGet(...) {  
        // Process Get  
    }  
  
    protected void doPut(...) {  
        // Process Put  
    }  
}
```





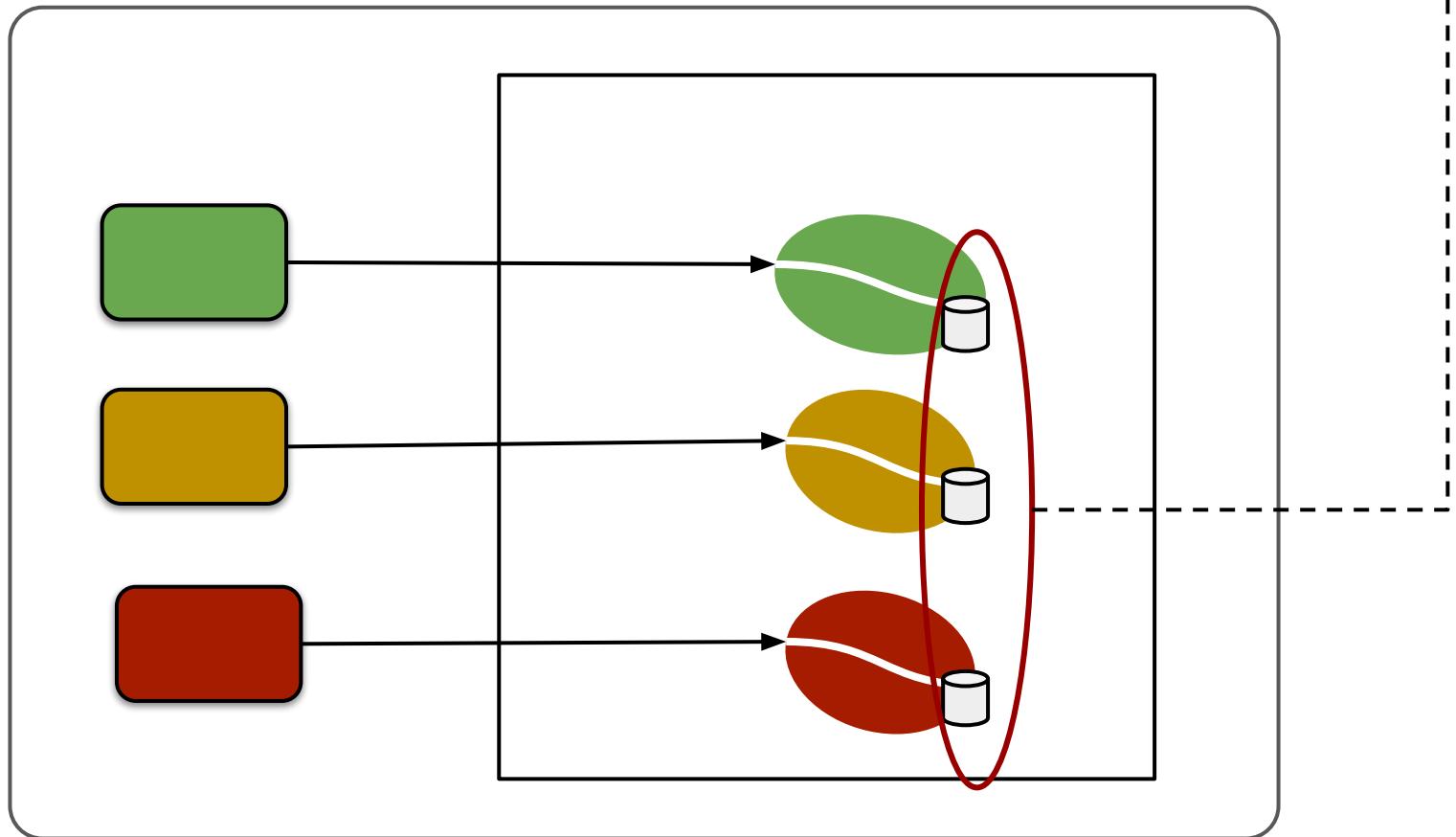
www.mem.com

Stateful



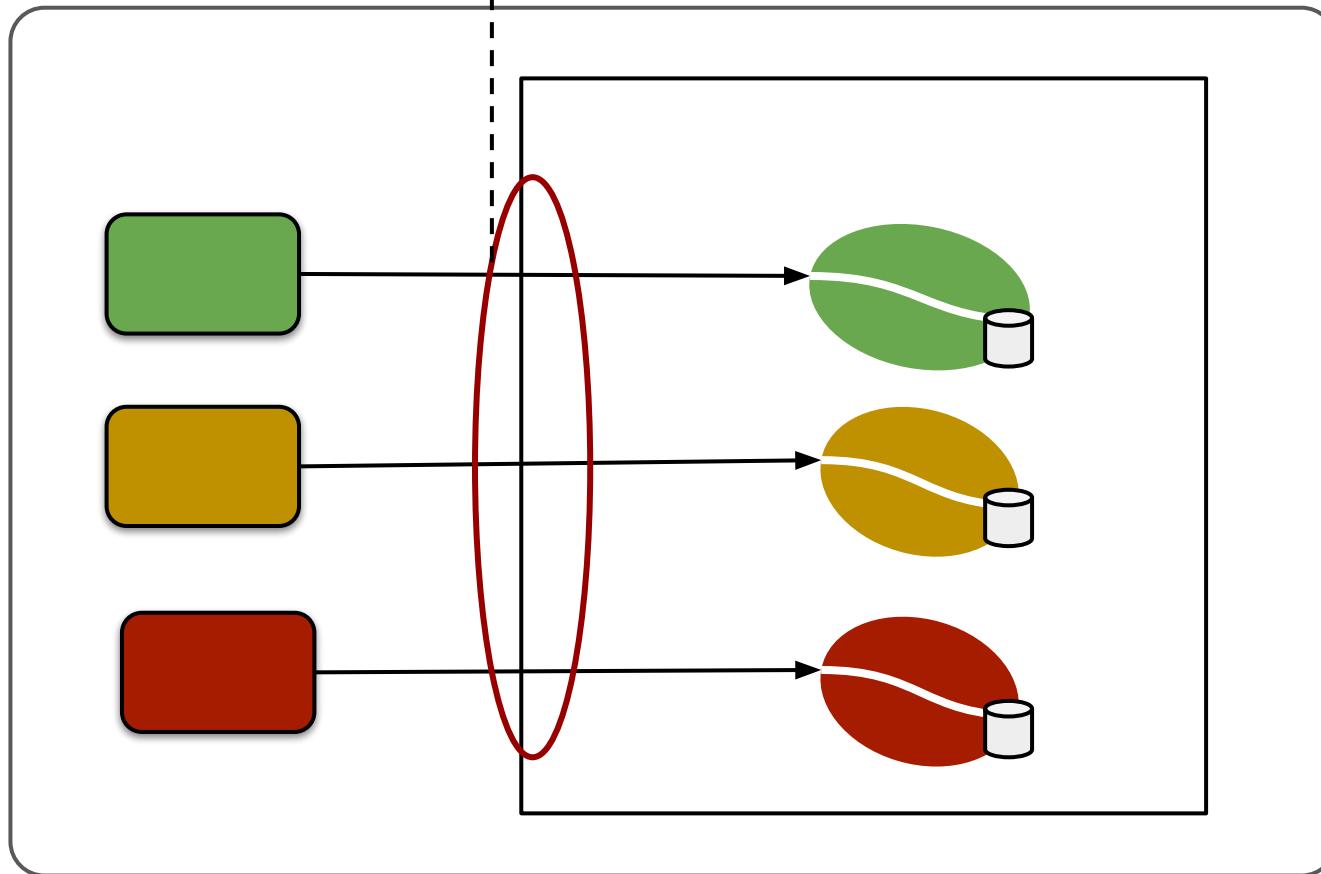
Stateful

Há estado de conversação ←-----



Stateful

-----> Um cliente, um bean



```
@Stateful(name = "MyBean",
           mappedName = "MyMappedName",
           description = "Desc",
           passivationCapable = true)
public class MyBusinessBean implements MyLocalInterface {

    private Map<String, Integer> counts = new HashMap();

    public void process(String msg) {
        counts.put(...);
    }

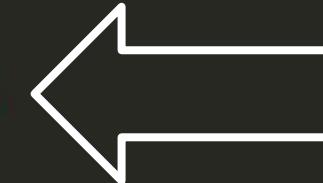
    @Remove public Map<String, Integer> counts() {
        /* Do something */
    }
}
```

```
@Stateful(name = "MyBean",
           mappedName = "MyMappedName",
           description = "Desc",
           passivationCapable = true)
public class MyBusinessBean implements MyLocalInterface {

    private Map<String, Integer> counts = new HashMap();

    public void process(String msg) {
        counts.put(...);
    }

    @Remove public Map<String, Integer> counts() {
        /* Do something */
    }
}
```



```
@Stateful(name = "MyBean",
           mappedName = "MyMappedName",
           description = "Desc",
           passivationCapable = true)
public class MyBusinessBean implements MyLocalInterface {

    private Map<String, Integer> counts = new HashMap();

    public void process(String msg) {
        counts.put(...);
    }

    @Remove public Map<String, Integer> counts() {
        /* Do something */
    }
}
```



```
@SessionScoped  
public class MyController {  
  
    @EJB private MyBusinessBean bean;  
  
    protected void doSomething() { }  
}
```

```
@SessionScoped  
public class MyController {  
    @EJB private MyBusinessBean bean;  
    protected void doSomething() {}  
}
```

SÓ PODE HAVER UM!

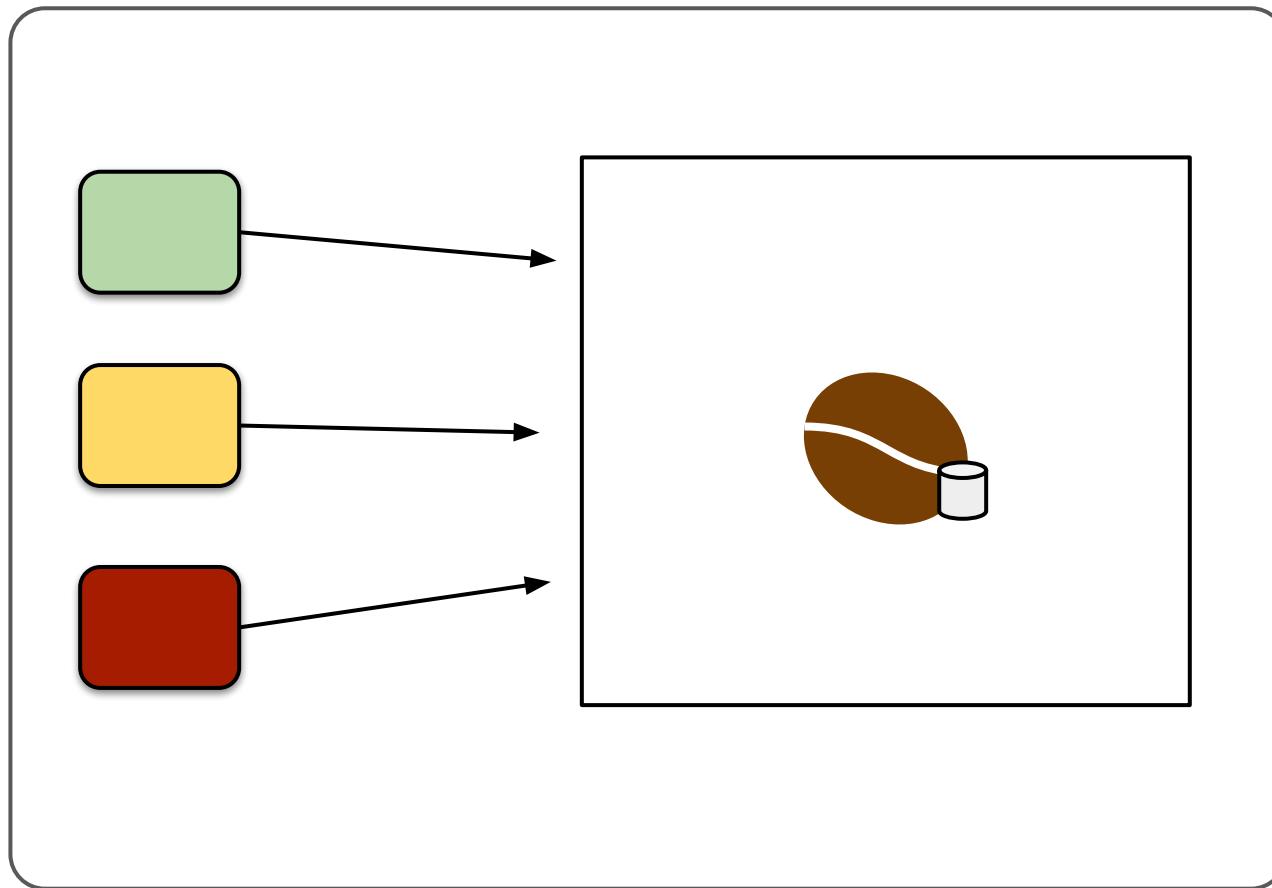
HIGHLANDER

O GUERREIRO IMORTAL

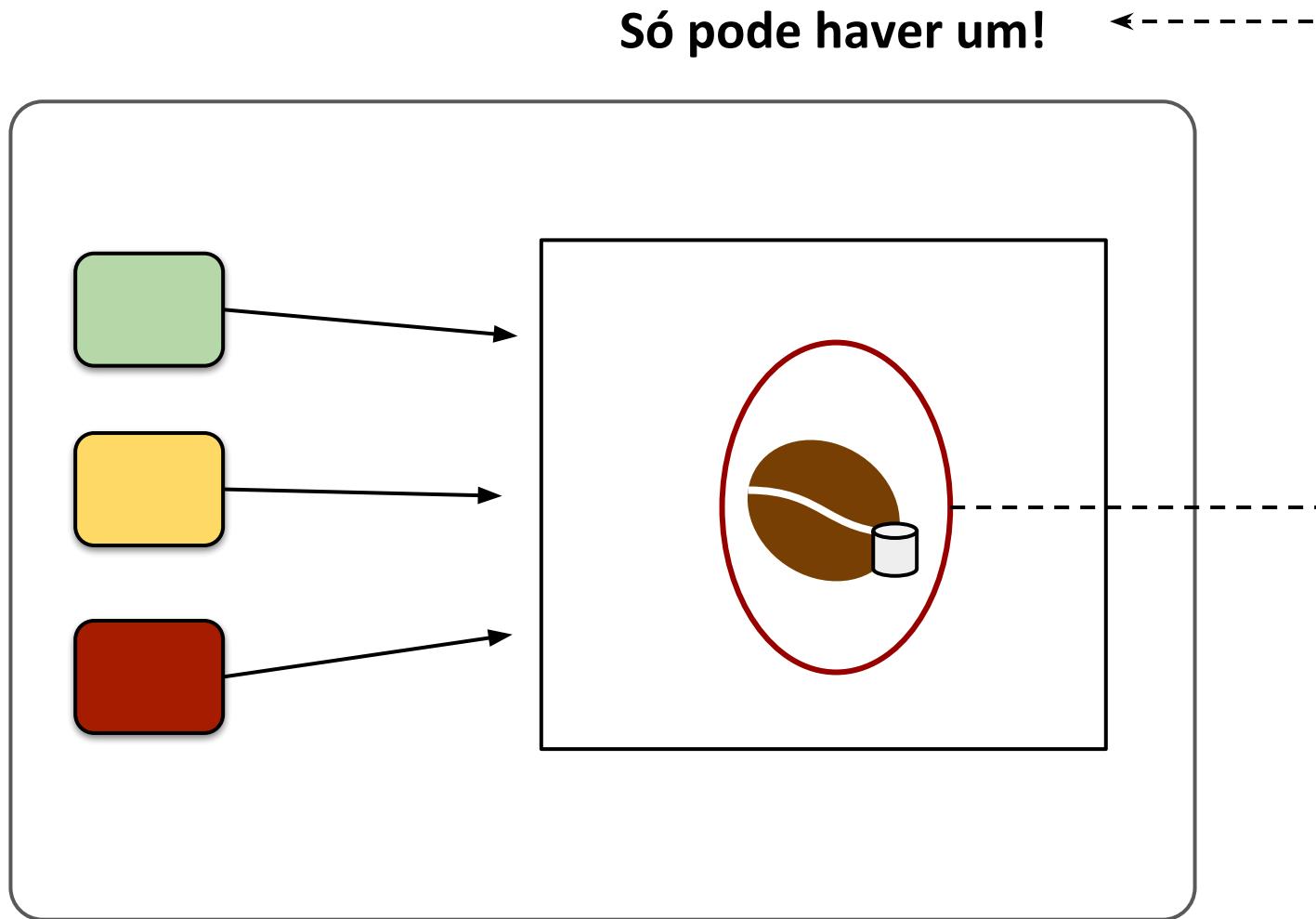
LOS INMORTALES
HIGHLANDER



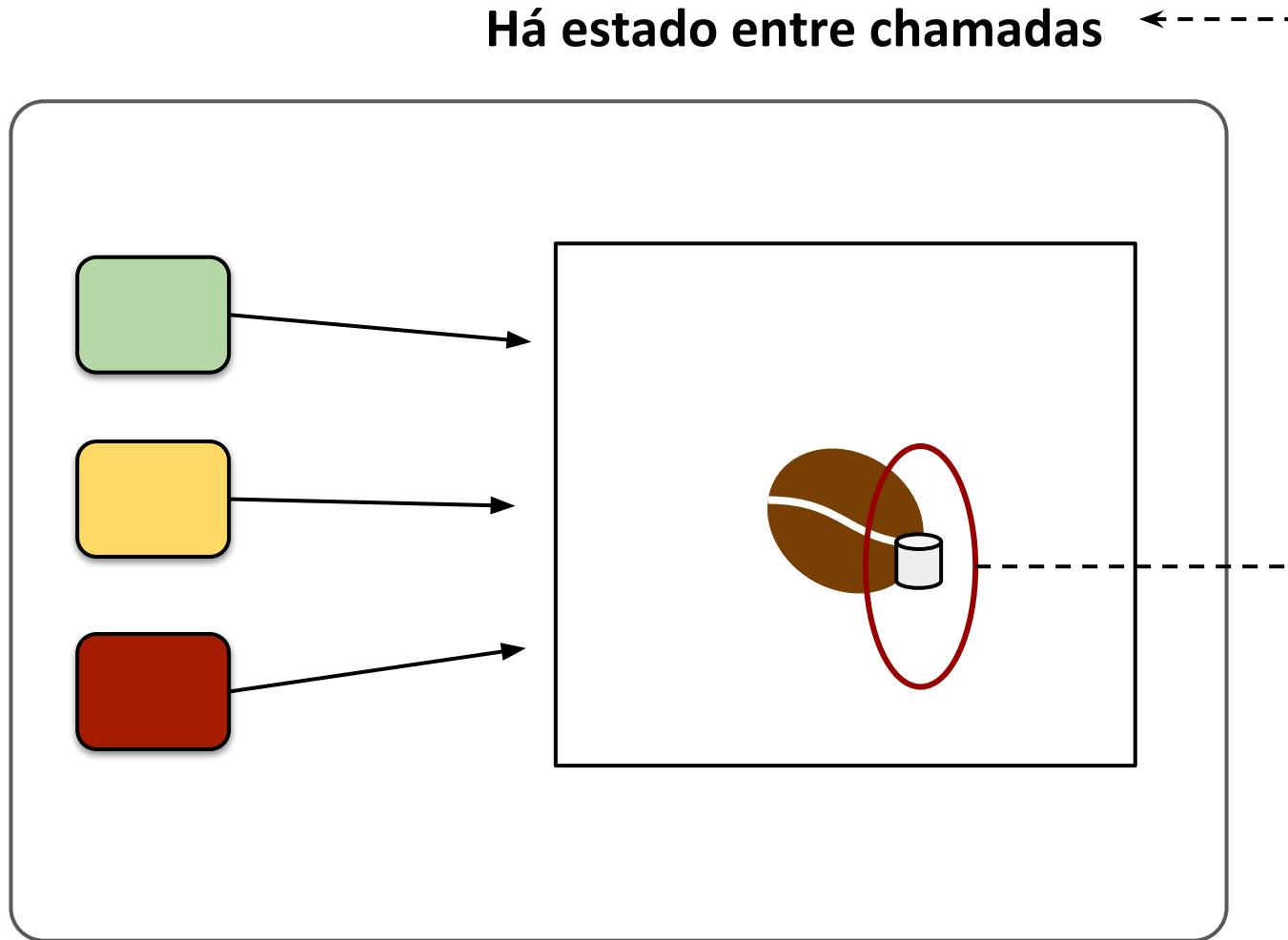
Singleton



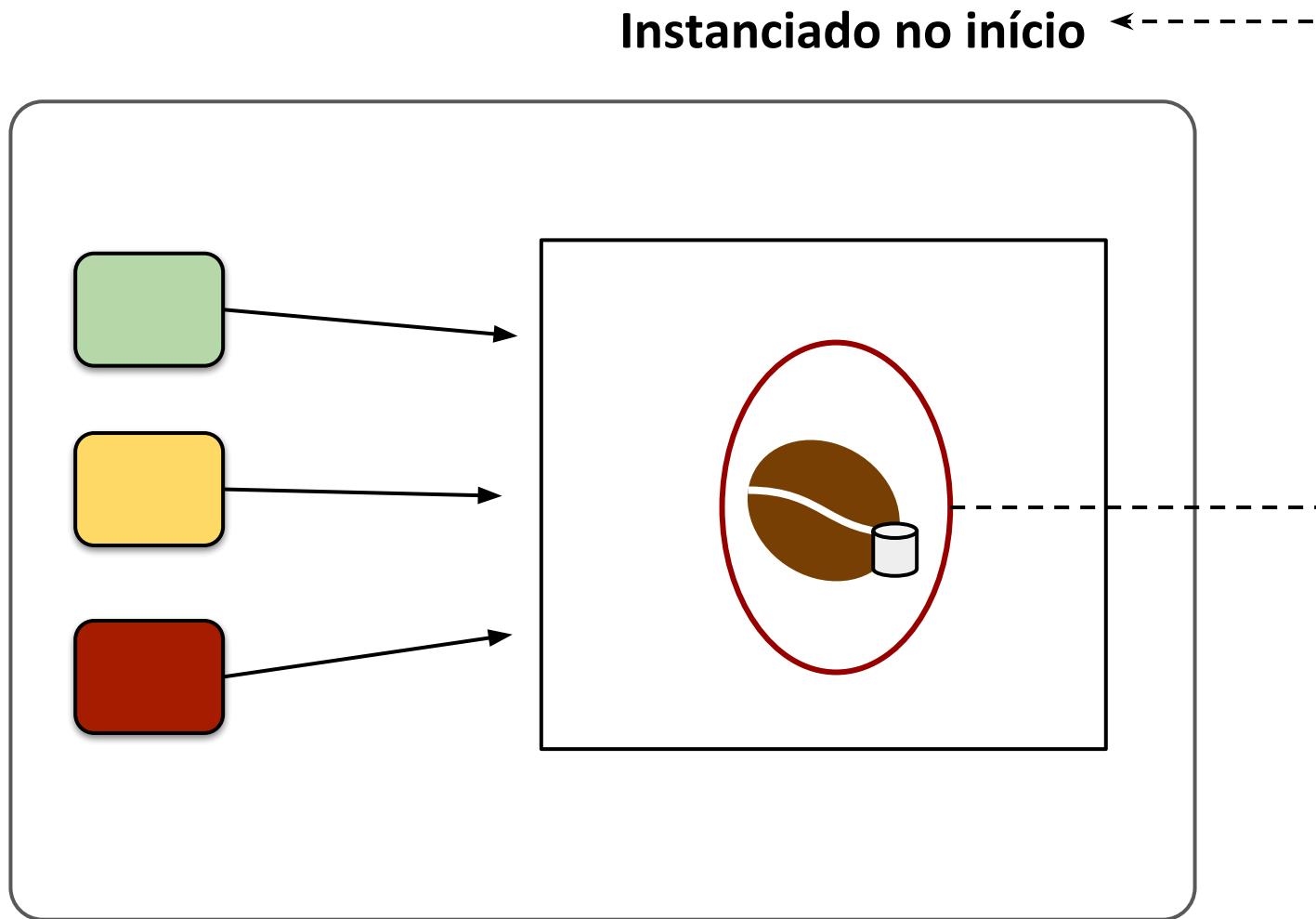
Singleton



Singleton



Singleton



```
@Singleton  
@Startup  
public class MyConfigurationsBean implements Configuration {  
    protected void doSomethingAtStartup() { }  
}
```

```
@Singleton  
@Startup  
public class MyConfigurationsBean implements Configuration {  
    protected void doSomethingAtStartup() { }  
}
```



Quando utilizar

Stateless Session Bean (SSSB)

Operações comuns: Verificar Estoque, debitar conta, etc

Stateful Session Bean (SSSB)

Estado: Wizards, Carrinhos de compra, etc

Singleton Session Bean (STSB)

Variável global, estado na camada de negócio, etc

Dicas

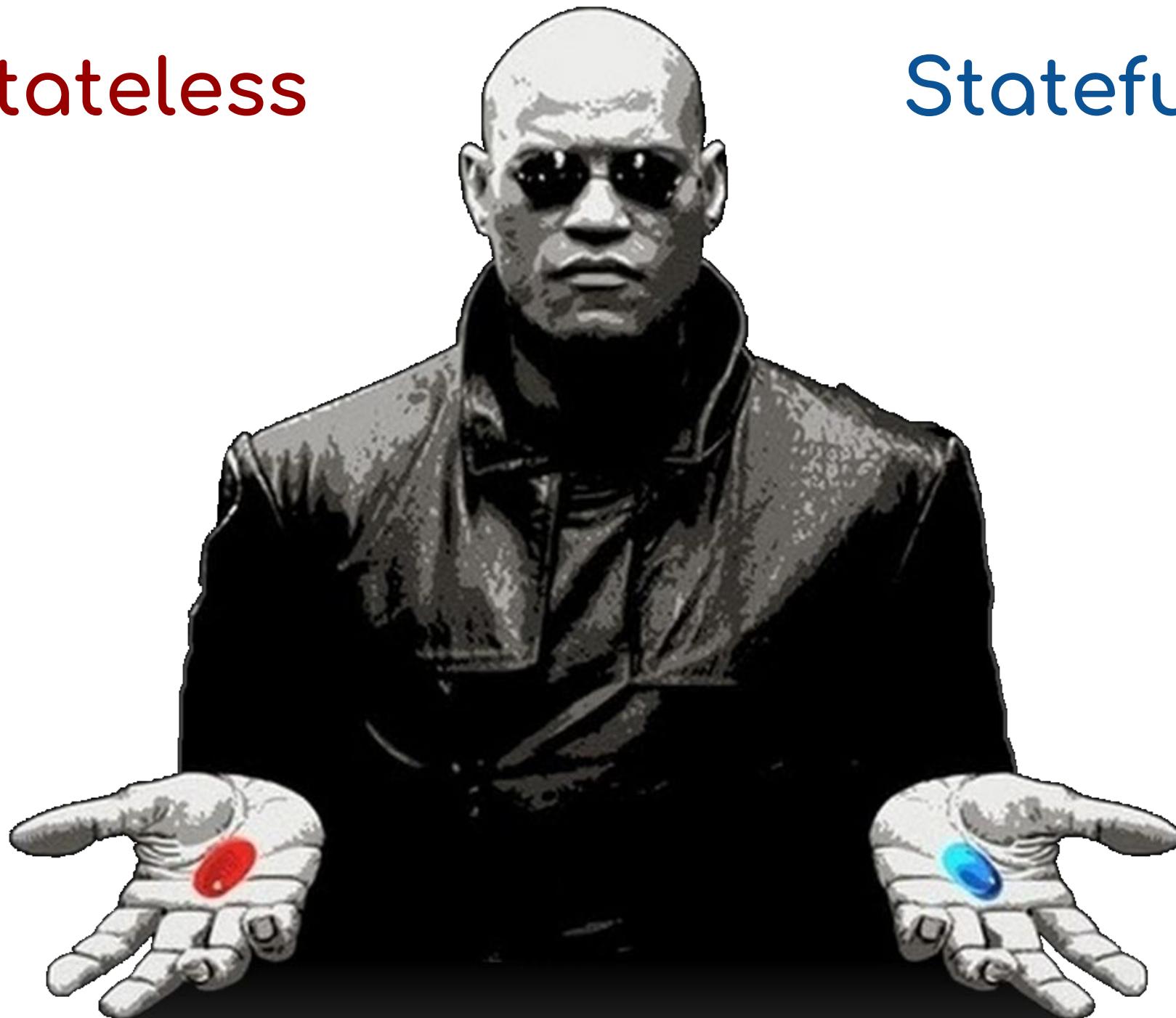
- Nunca confie nos atributos de classe de um SSSB
- Não dá para injetar SFSB em elementos *stateless*
- O STSB também serve para executar ações no “fim” da aplicação

Stateless vs Stateful??



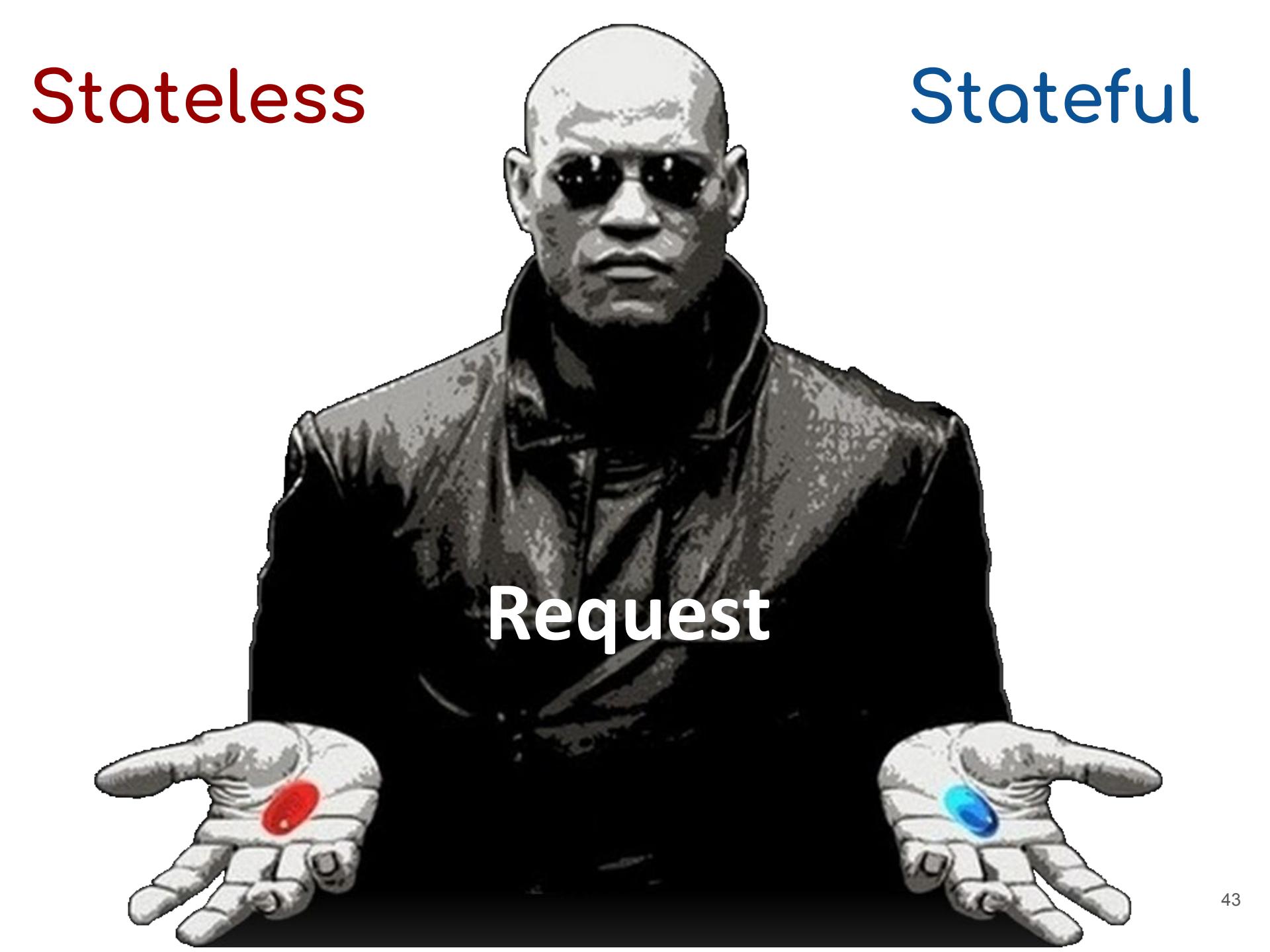
Stateless

Stateful



Stateless

Stateful



Request

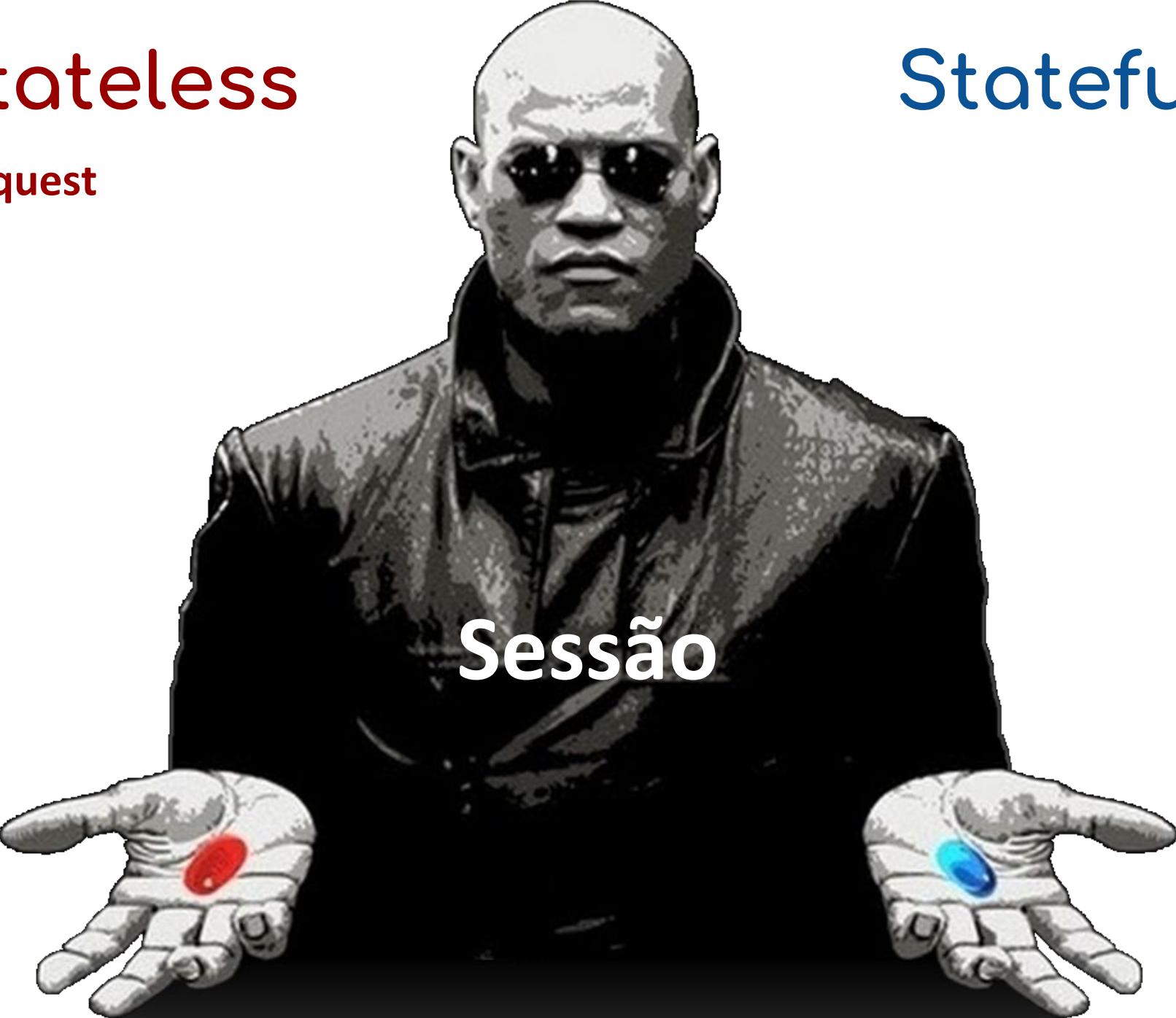
Stateless Request



Stateful

Stateless
Request

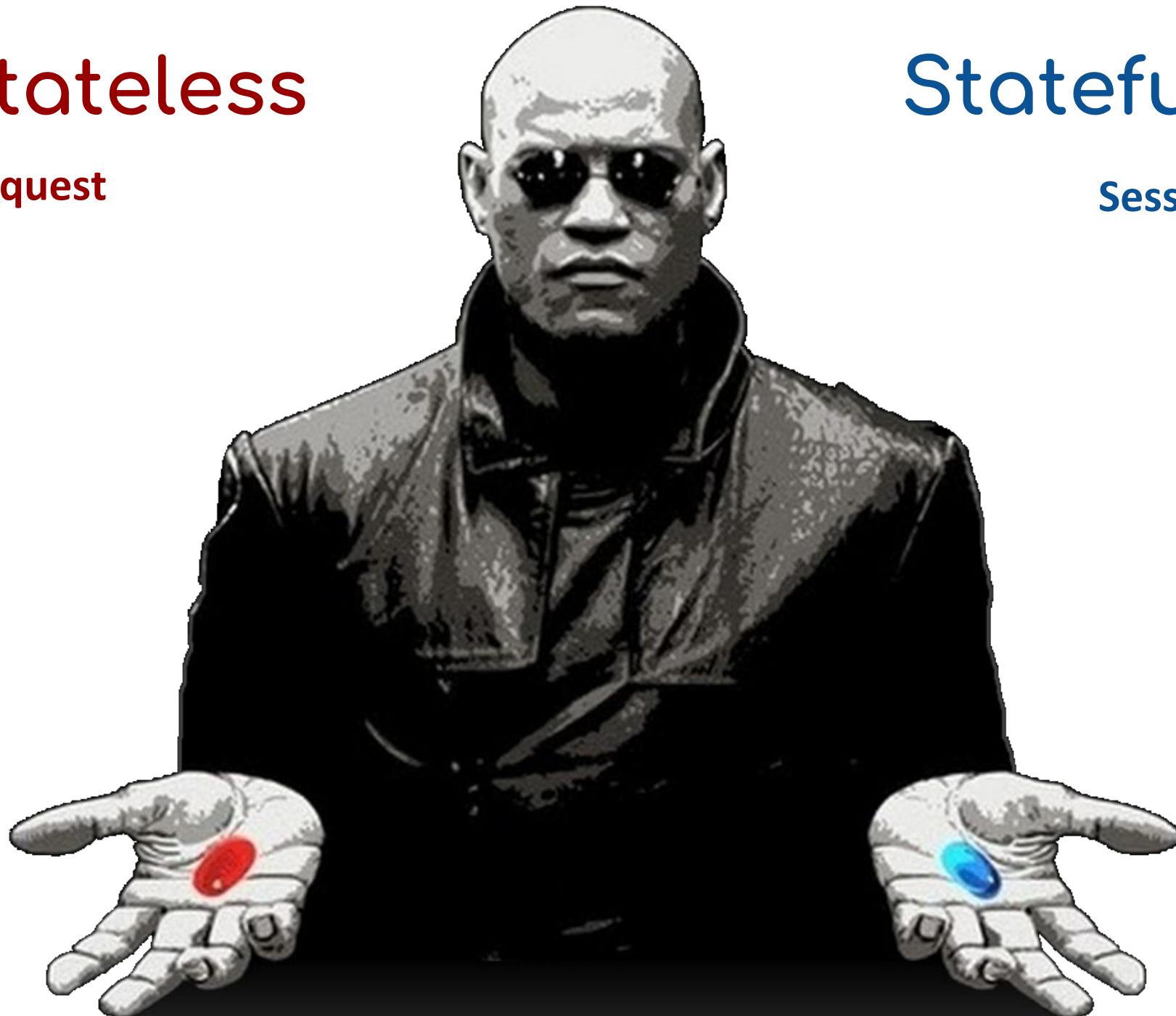
Stateful



Sessão

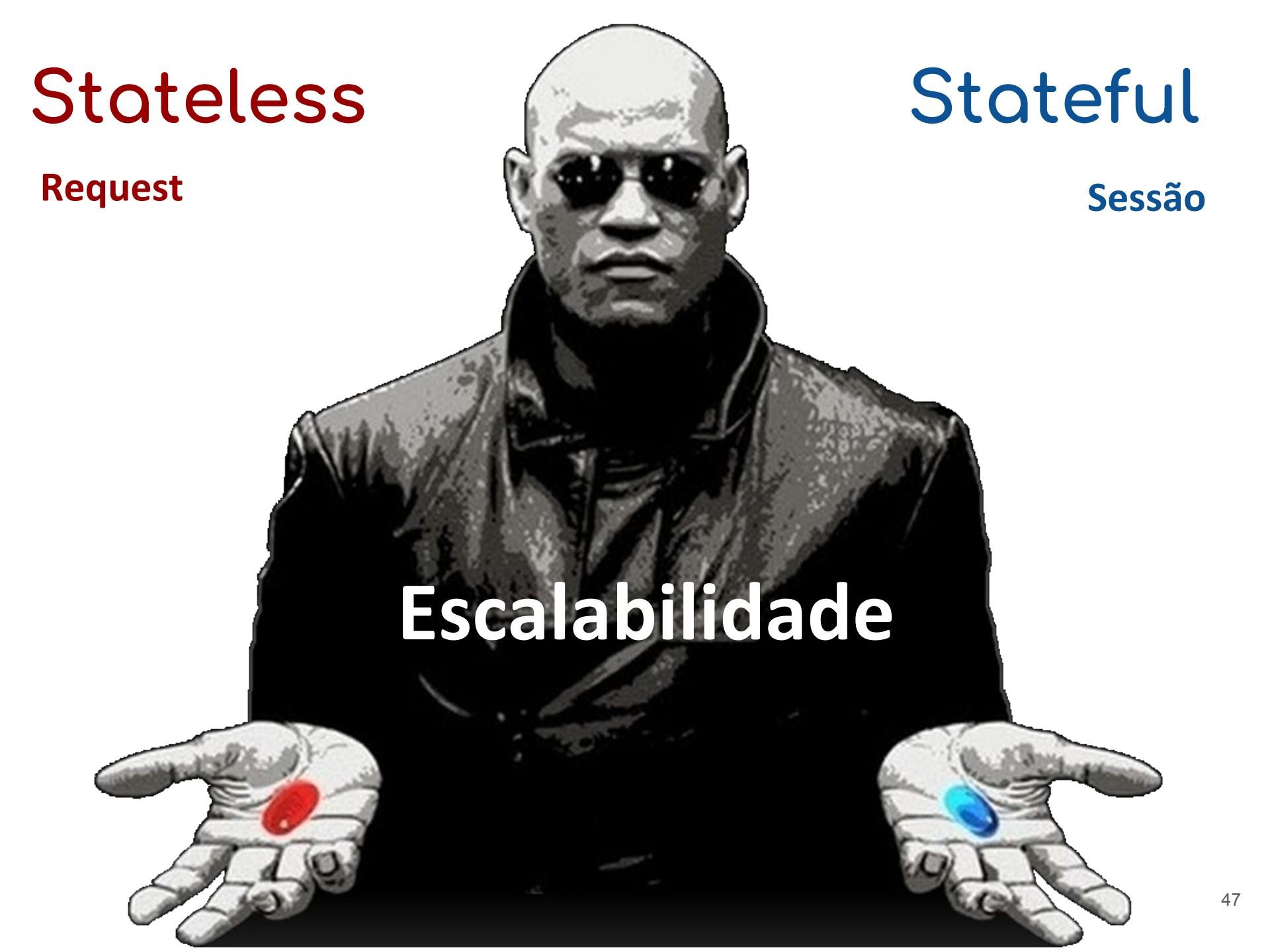
Stateless Request

Stateful Sessão



Stateless
Request

Stateful
Sessão



Escalabilidade

Stateless

Request

Escalabilidade

Stateful

Sessão



Stateless

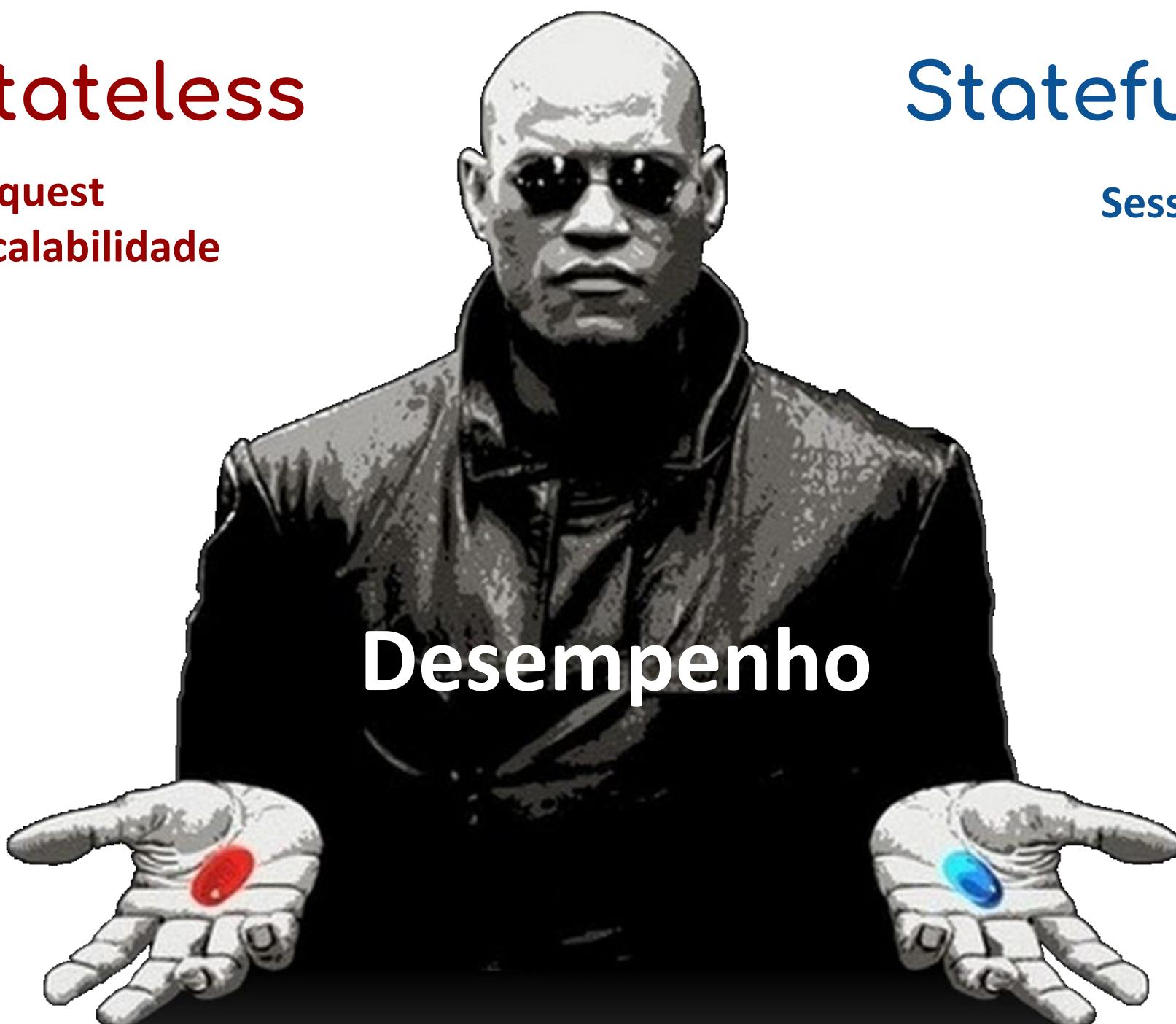
Request

Escalabilidade

Stateful

Sessão

Desempenho



Stateless

Request

Escalabilidade

Desempenho

Stateful

Sessão



Stateless

Request

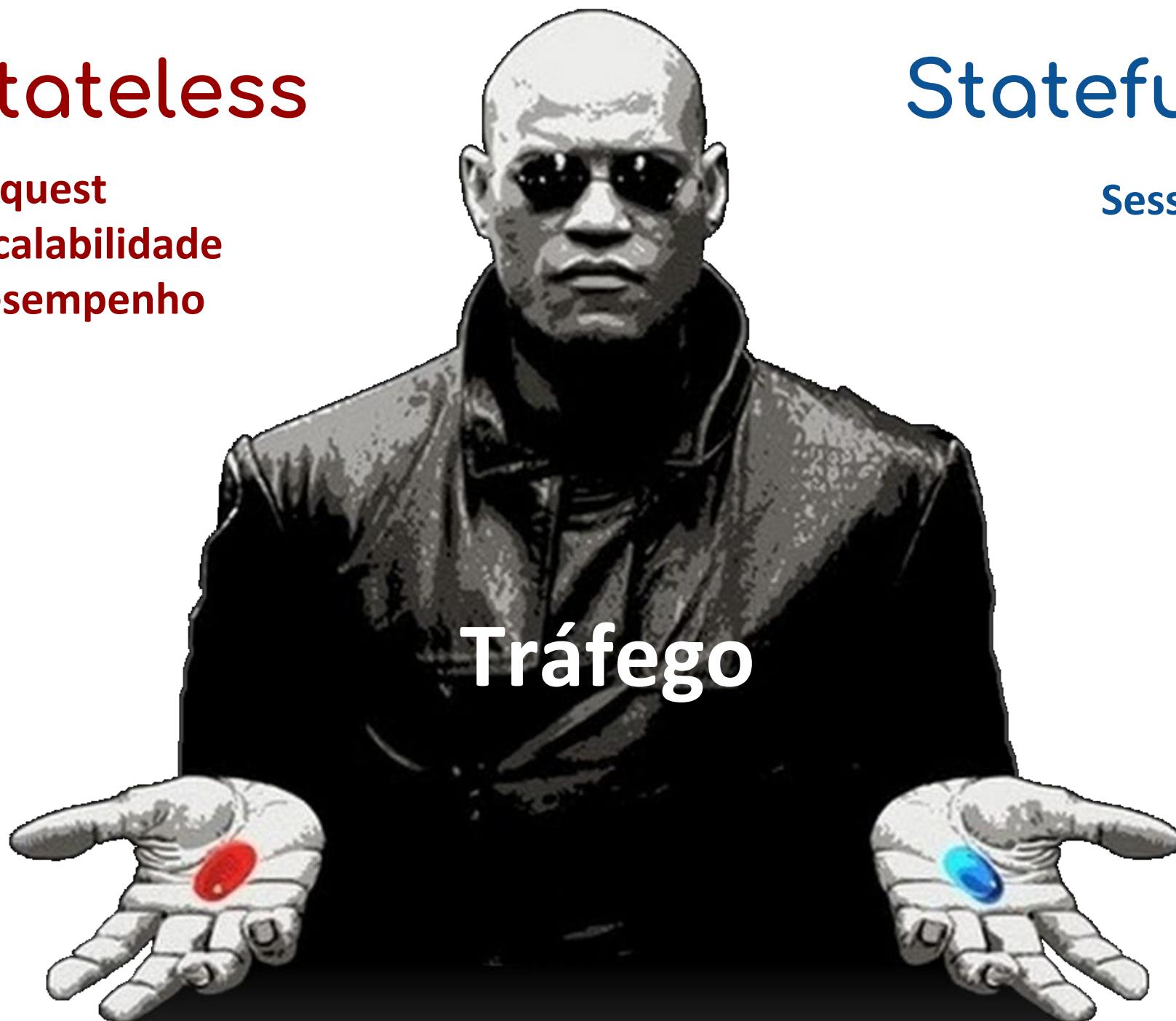
Escalabilidade

Desempenho

Stateful

Sessão

Tráfego



Stateless

Request

Escalabilidade

Desempenho

Stateful

Sessão
Tráfego



Stateless

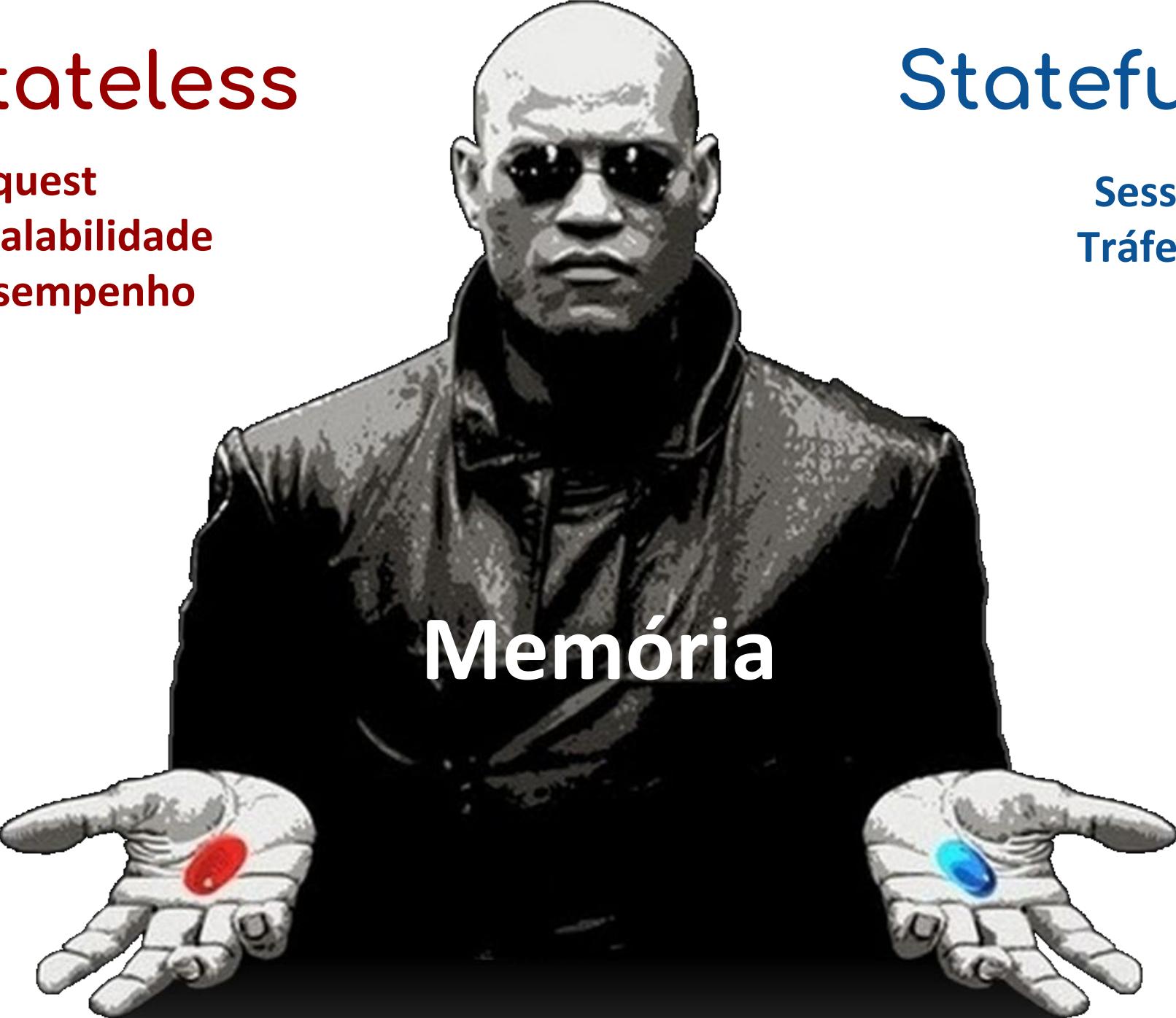
Request

Escalabilidade

Desempenho

Stateful

Sessão
Tráfego



Memória

The central figure is a black and white illustration of Neo from The Matrix. He is wearing sunglasses and a dark, textured jacket. He is holding two small, glowing spheres in his hands, one red in his left hand and one blue in his right hand. The word "Memória" is overlaid in large, white, sans-serif font across the center of the image.

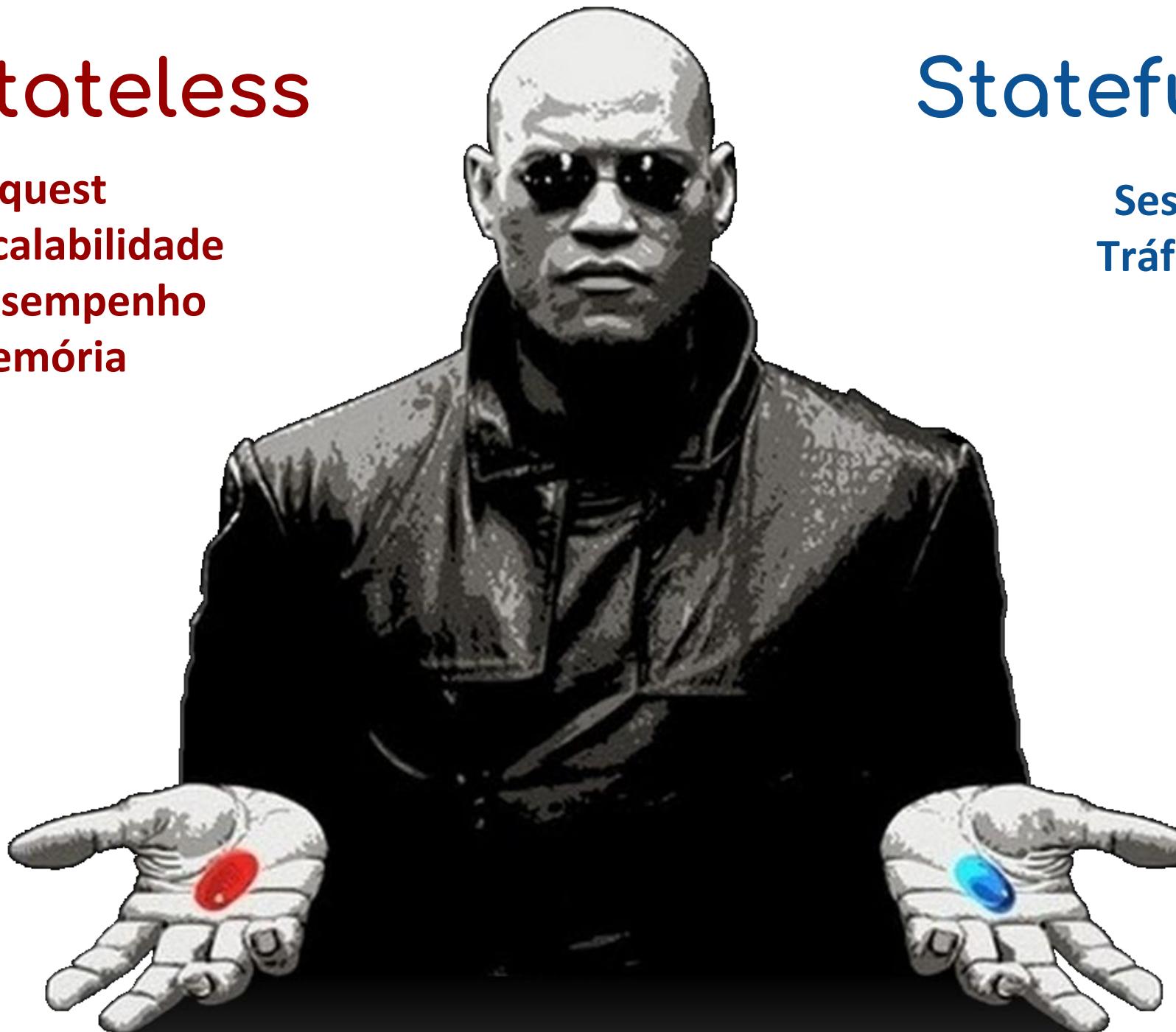
Stateless

Request

Escalabilidade

Desempenho

Memória

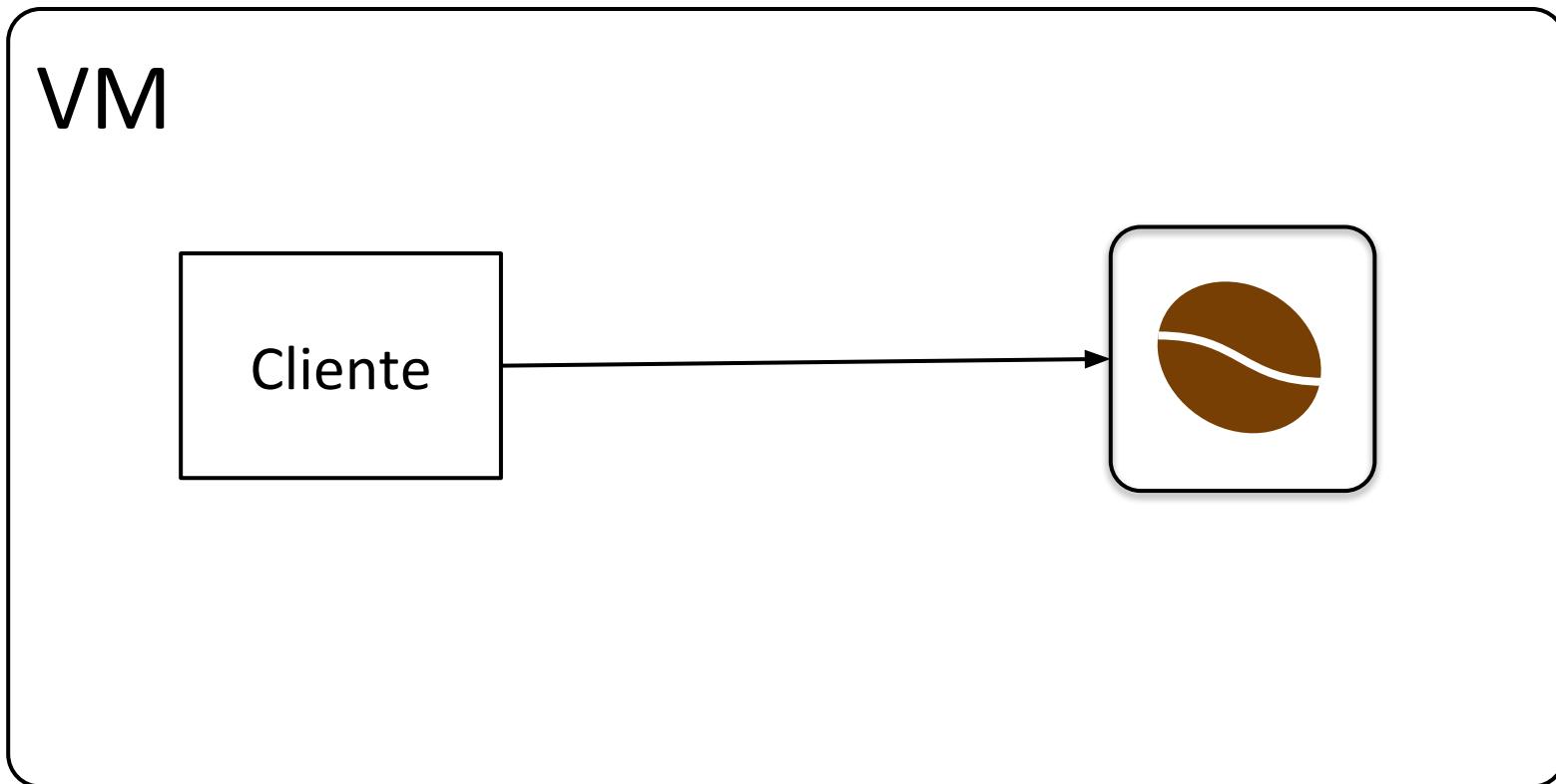


Stateful

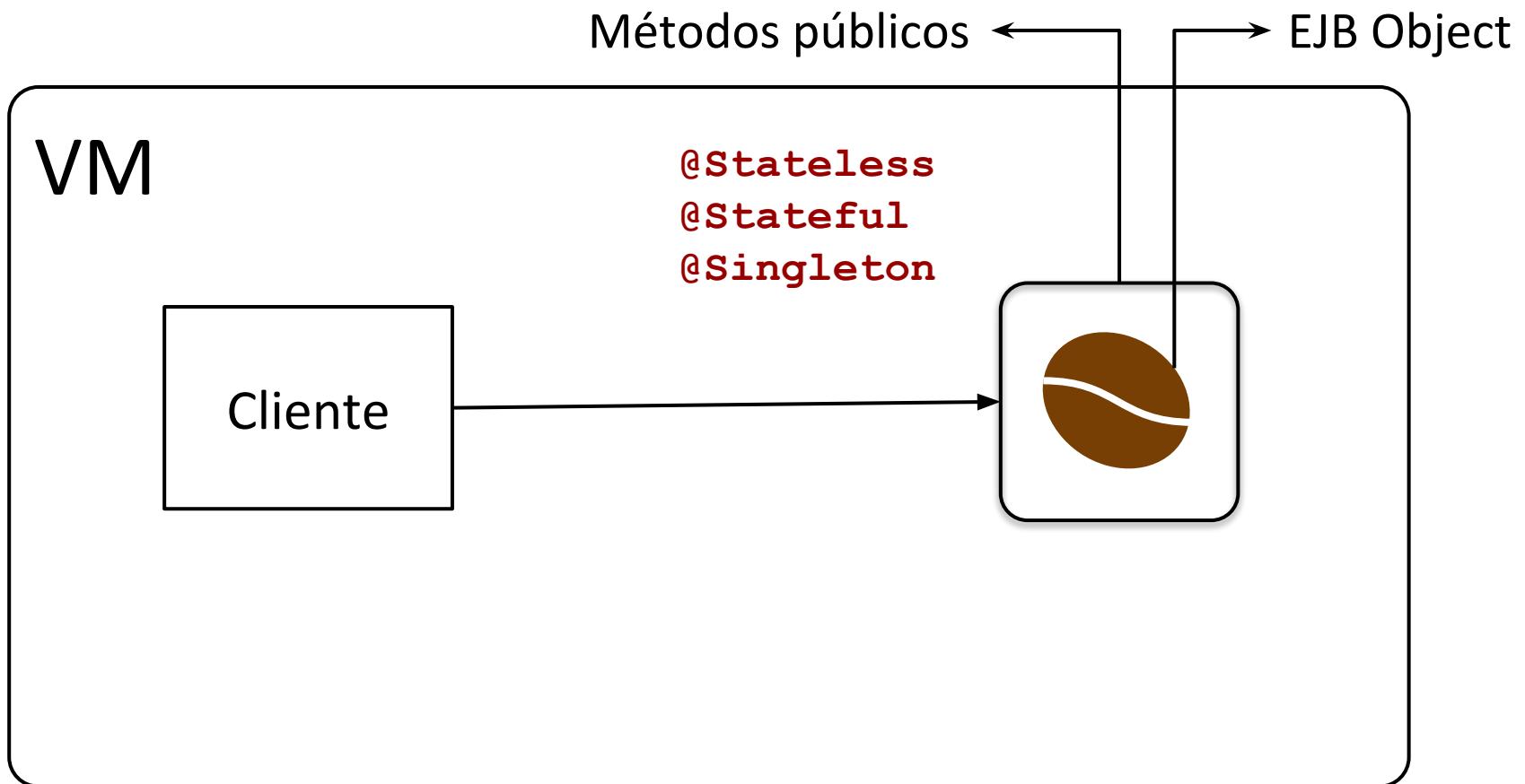
Sessão
Tráfego

Interfaces e Visão do Cliente

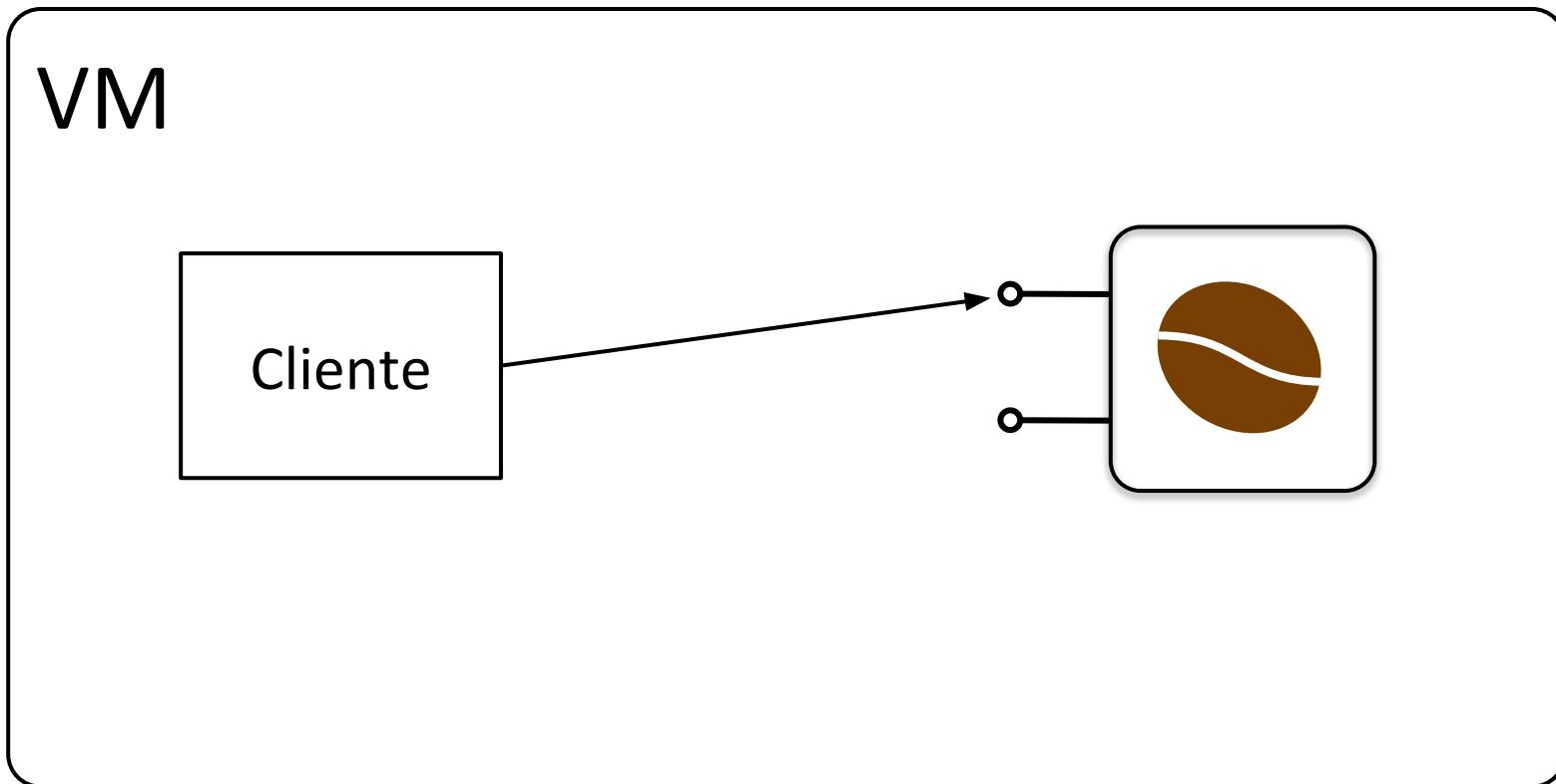
Visão sem Interface



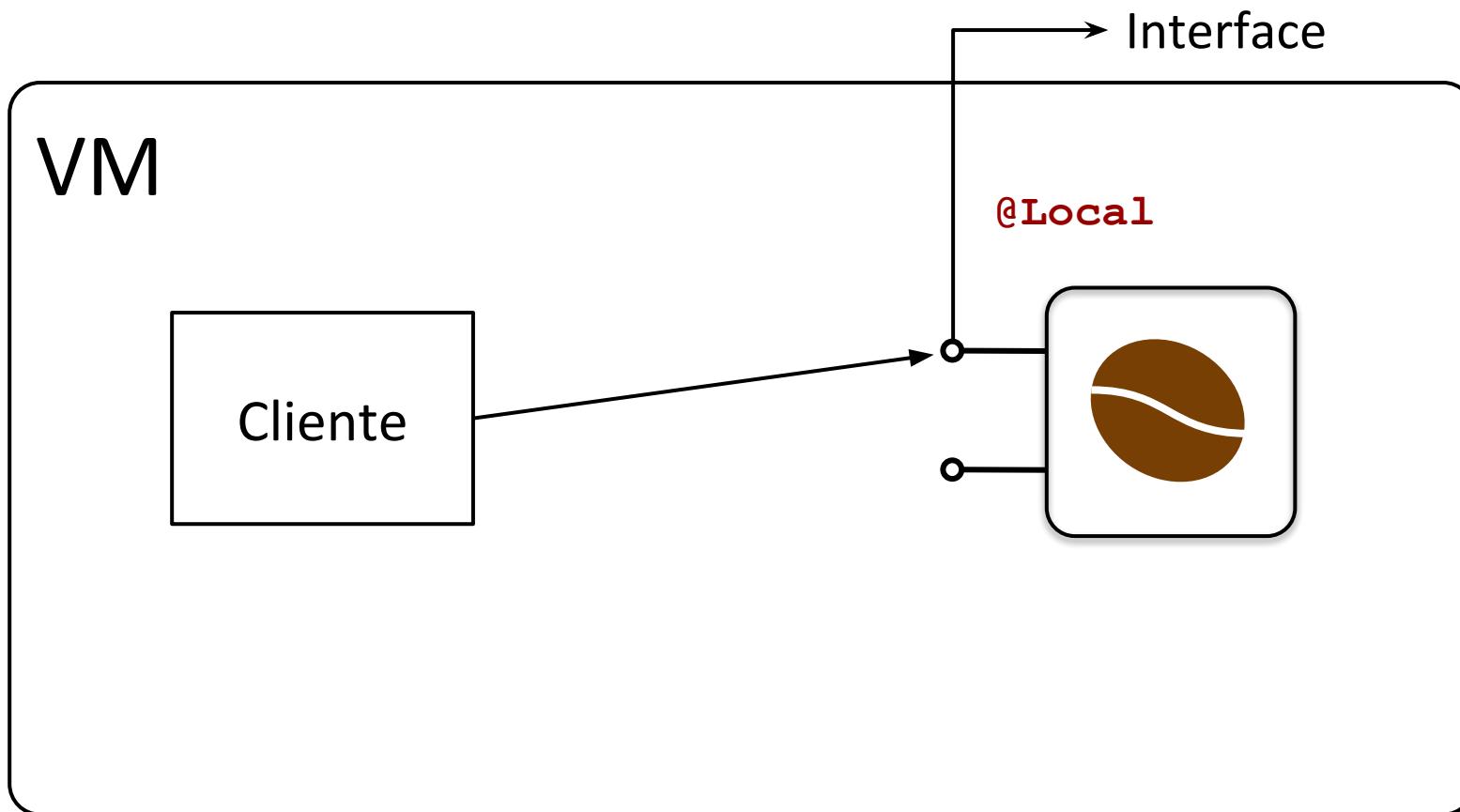
Visão sem Interface



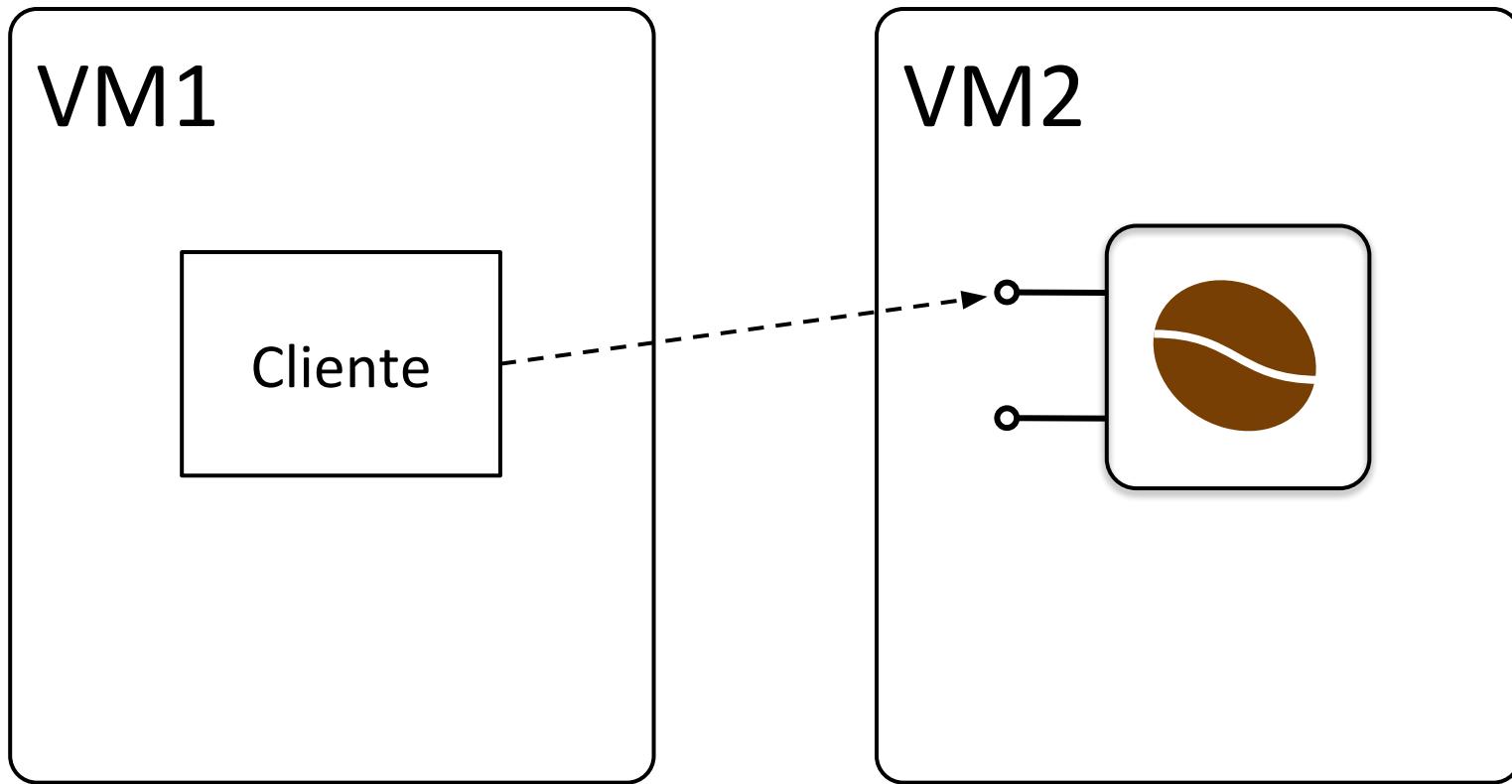
Visão Local



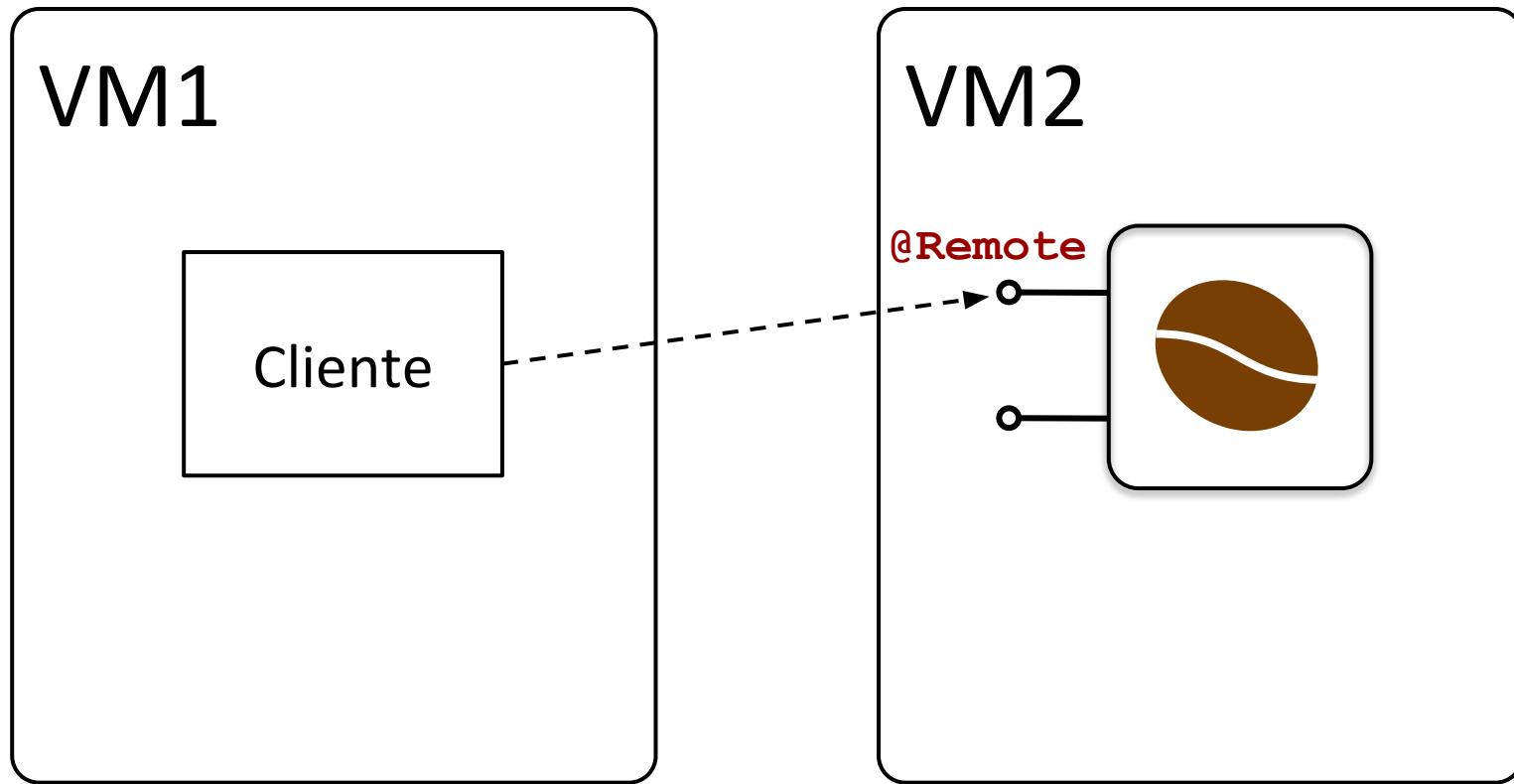
Visão Local



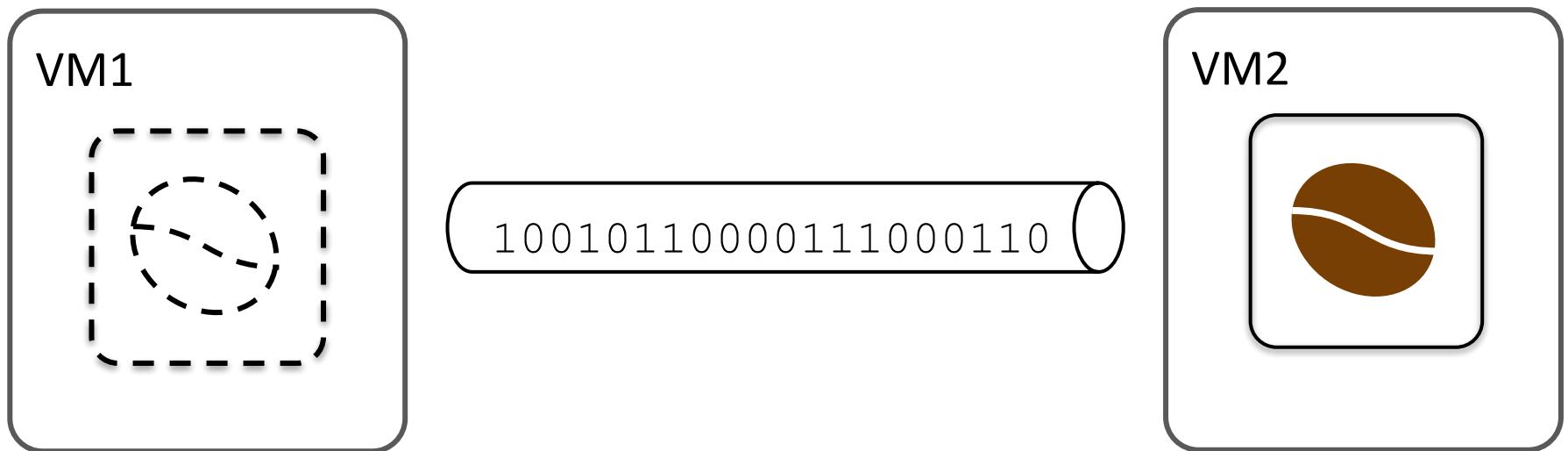
Visão Remota



Visão Remota



Serialize Já!



Obtendo referência

```
1 @RequestScoped
2 public class MyClient {
3
4     @EJB private MyEjb control;
5
6     public void doSomething() {
7         control.process();
8     }
9 }
```

Obtendo referência

```
1 @RequestScoped
2 public class MyClient {
3
4     @EJB private MyEjb control;
5
6     public void doSomething() {
7         control.process();
8     }
9 }
```



Obtendo referência

```
1 @RequestScoped
2 public class MyClient {
3
4     @Resource SessionContext ctx;
5     private MyEjb control;
6
7     public void doSomething() {
8         control = (MyEjb)ctx.lookup("MeuEJB");
9         control.process();
10    }
11 }
```

Obtendo referência

```
1 @RequestScoped
2 public class MyClient {
3
4     @Resource SessionContext ctx;
5     private MyEjb control;
6
7     public void doSomething() {
8         control = (MyEjb)ctx.lookup("MeuEJB");
9         control.process();
10    }
11 }
```



Obtendo referência

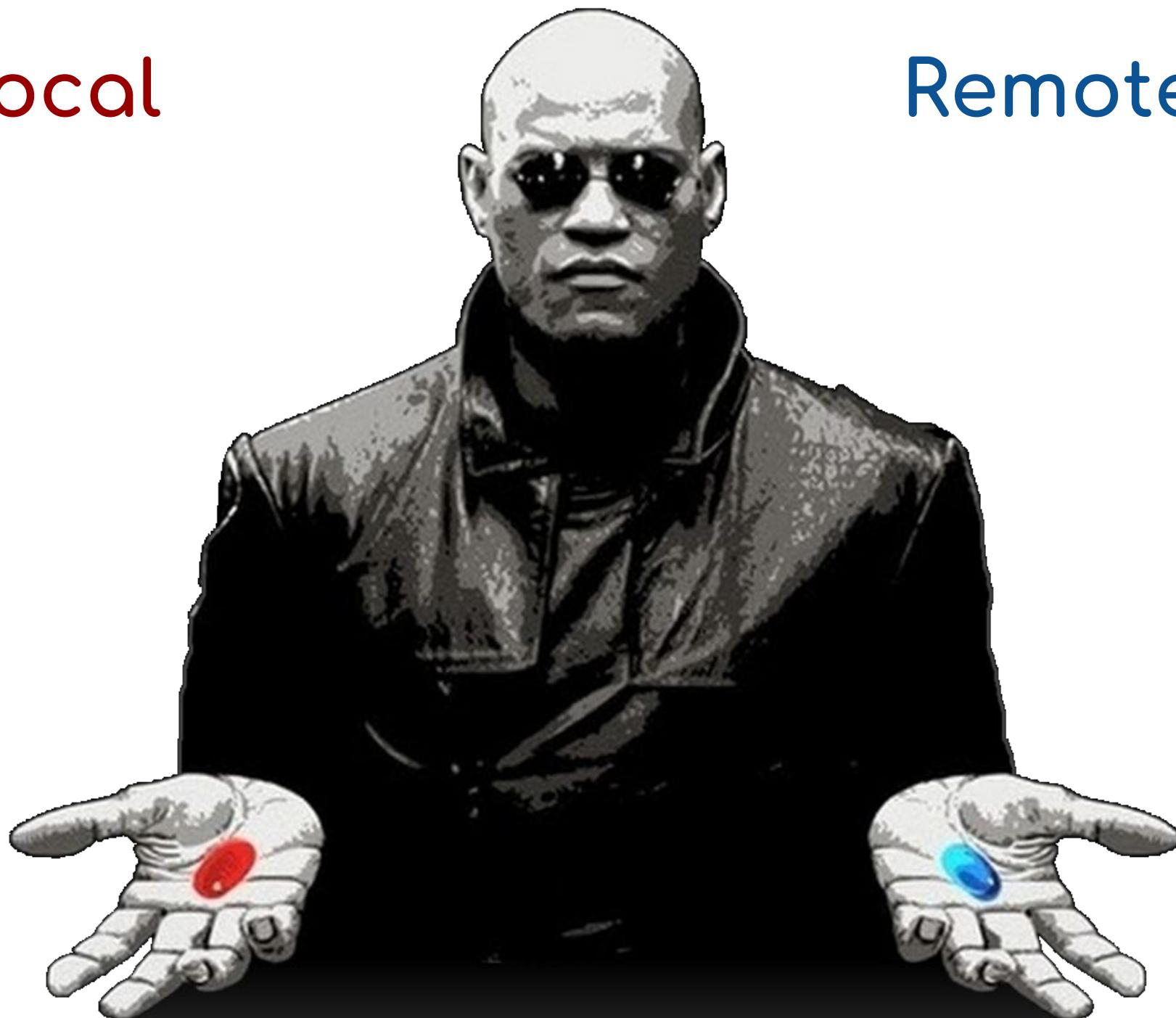
```
1 @RequestScoped
2 public class MyClient {
3
4     @Resource SessionContext ctx;
5     private MyEjb control;
6
7     public void doSomething() {
8         control = new MyEjb(...);
9         control.process();
10    }
11 }
```

Obtendo referência

```
1 @RequestScoped
2 public class MyClient {
3
4     @Resource SessionContext ctx;
5     private MyEjb control;
6
7     public void doSomething() {
8         control = new MyEjb(...);
9         control.process();
10    }
11 }
```



Local



Local

Remote

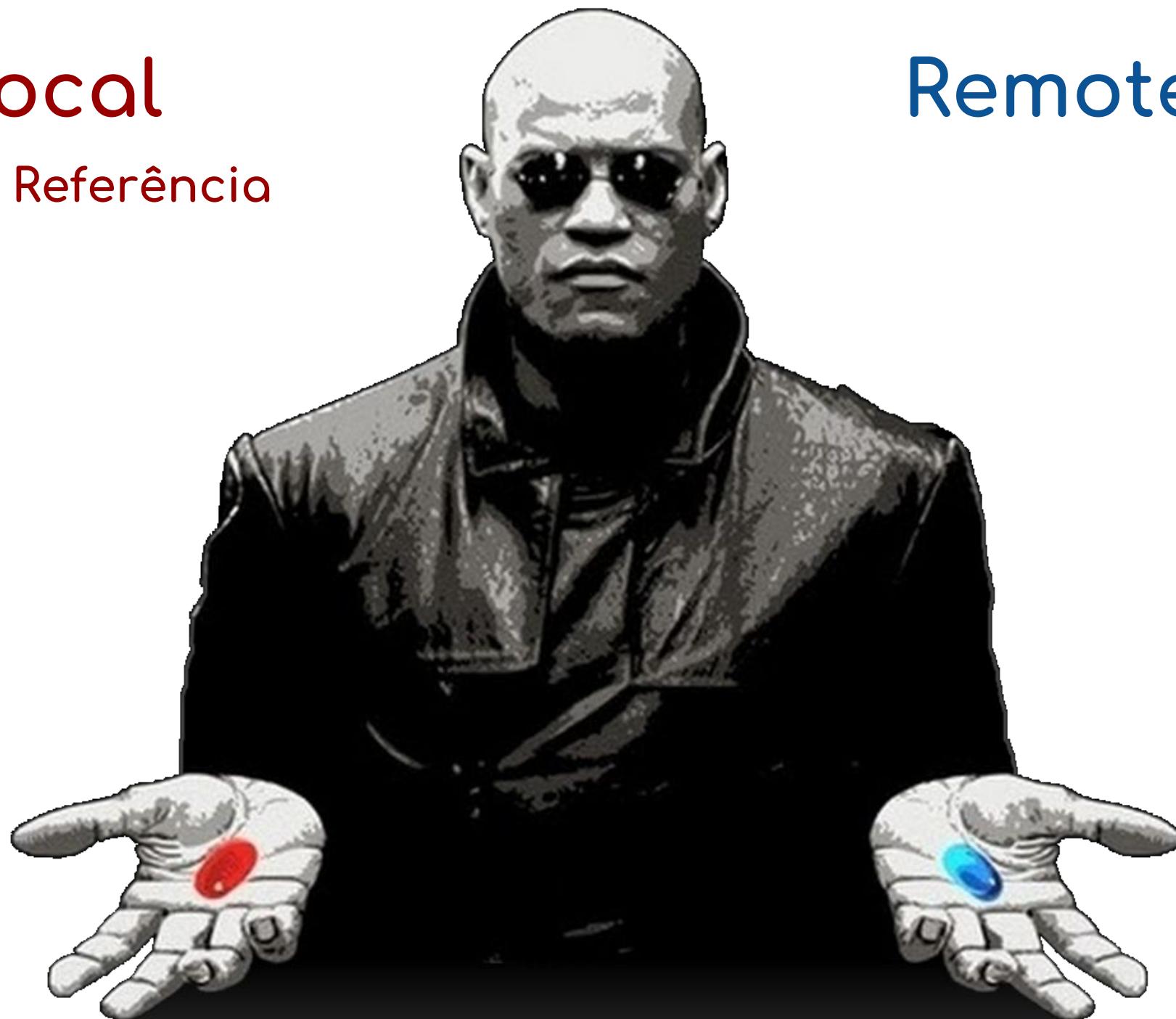


Chamada por
Referência

Local

C. Referência

Remote



Local

C. Referência

Remote



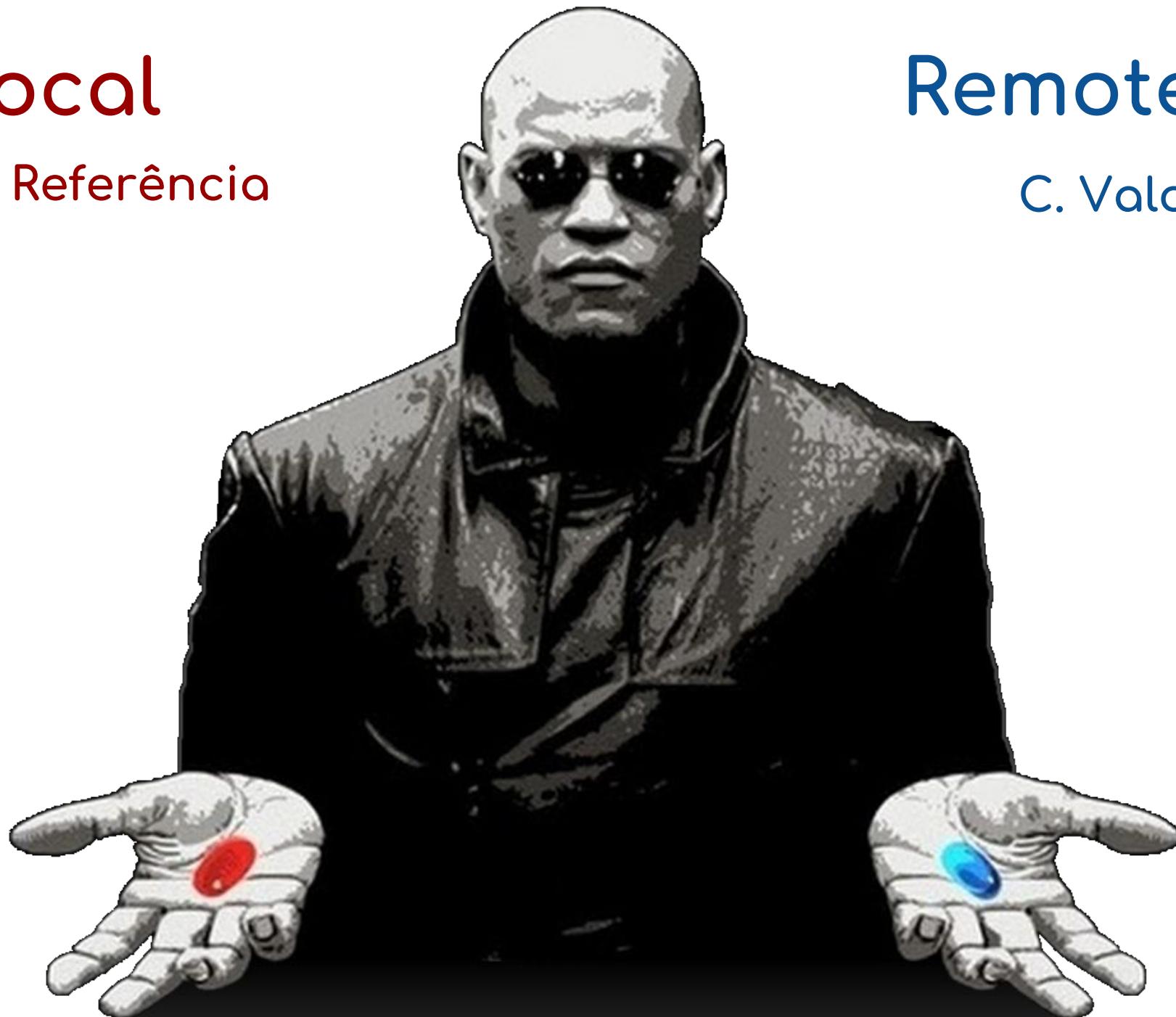
Chamada por
Valor

Local

C. Referência

Remote

C. Valor



Local

C. Referência

Remote

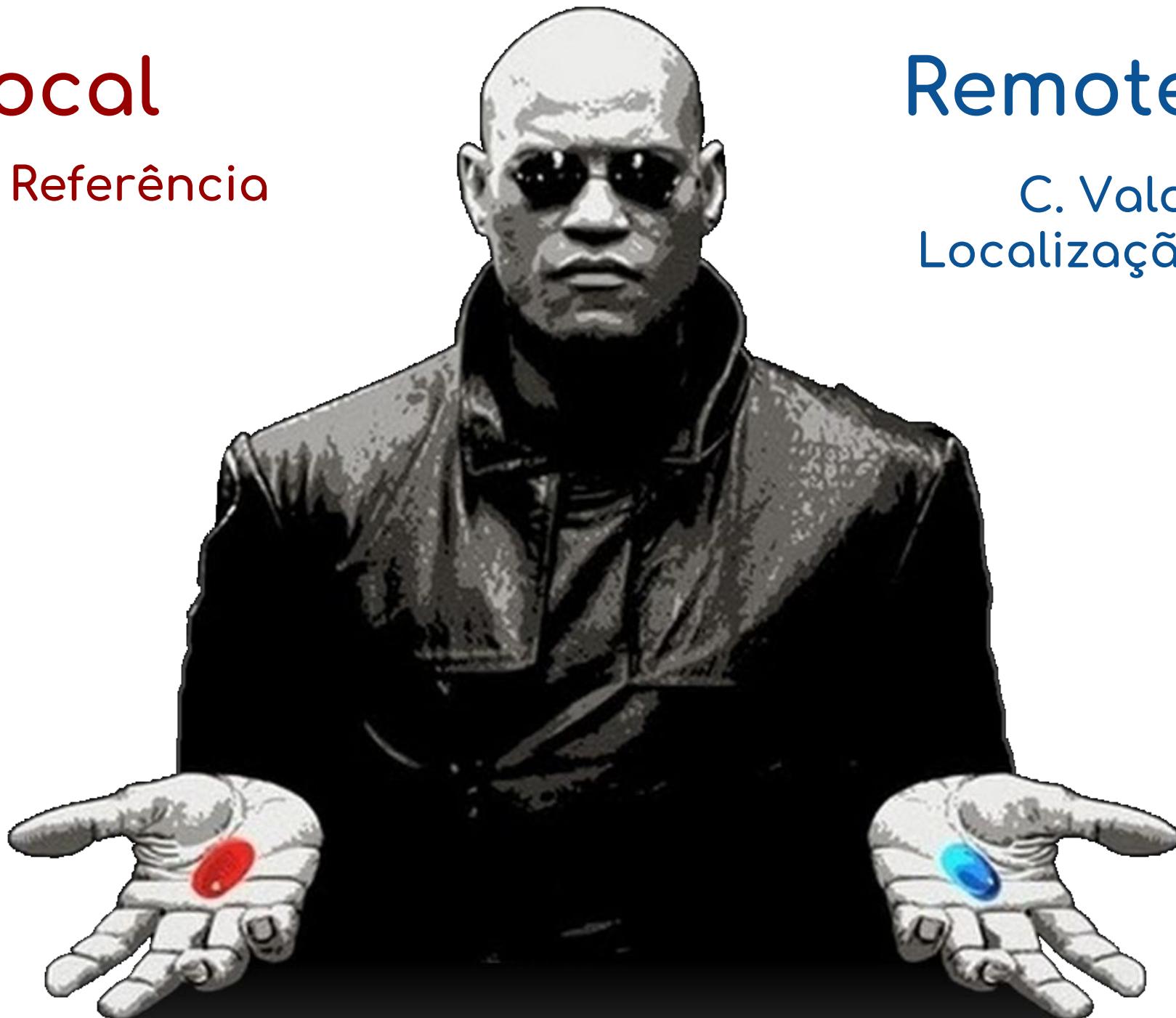
C. Valor

A black and white illustration of Neo from The Matrix. He is wearing his signature dark sunglasses and a black leather trench coat. His hands are held out in front of him, palms up. In his left hand, there is a red, glowing, teardrop-shaped object. In his right hand, there is a blue, glowing, teardrop-shaped object. The word "Localização" is overlaid in large, white, sans-serif text across the center of the image.

Localizaçāo

Local

C. Referência



Remote

C. Valor
Localização

Local

C. Referência

Remote

C. Valor
Localização

Rede

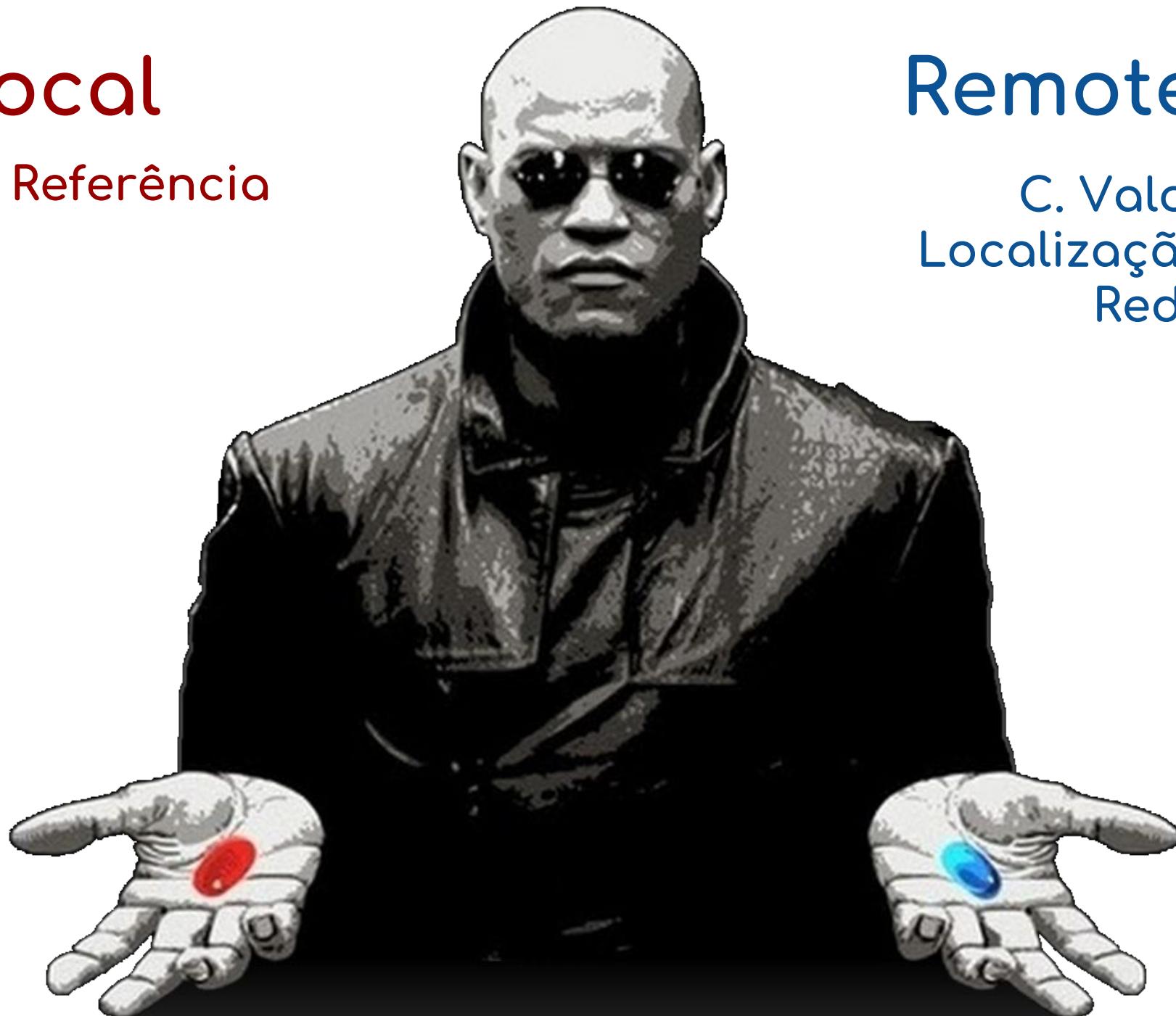


Local

C. Referência

Remote

C. Valor
Localização
Rede



Local

C. Referência

Remote

C. Valor

Localização
Rede



Granularidade
Fina

Local

C. Referência
G. Fina

Remote

C. Valor
Localização
Rede



**Granularidade
Grossa**

Local

C. Referência
G. Fina

Remote

C. Valor
Localização
Rede
G. Grossa



E a visão sem interfaces?!?!



Ciclo de Vida

Stateless \ Singleton

`@PostConstruct`
`@PreDestroy`

- ① Dependency injection, if any
- ② PostConstruct callback, if any

Does Not Exist

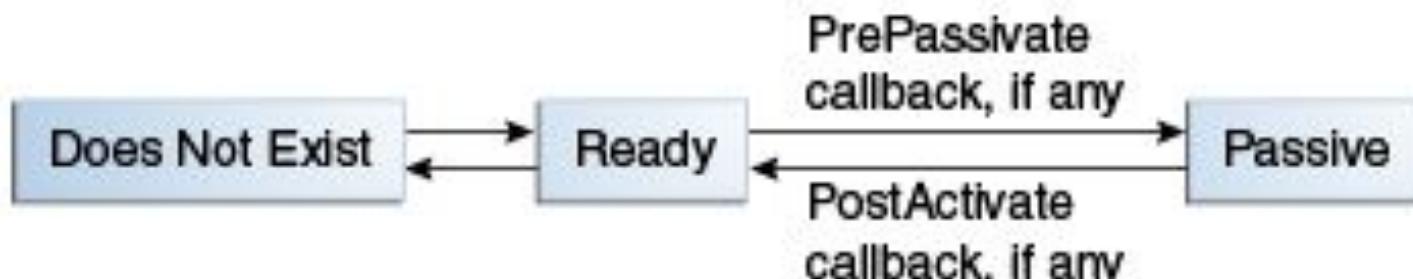
Ready

PreDestroy callback, if any

Stateful

- ① Create
- ② Dependency injection, if any
- ③ PostConstruct callback, if any
- ④ Init method, or ejbCreate<METHOD>, if any

@PostConstruct
@PreDestroy
@PrePassivate
@PostActivate

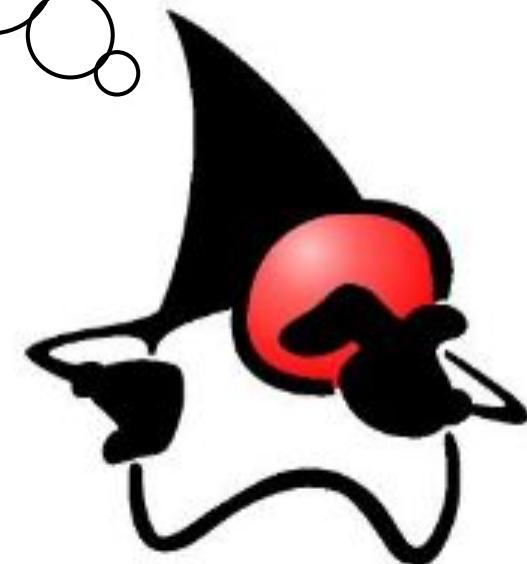


- ① Remove
- ② PreDestroy callback, if any

Stateful

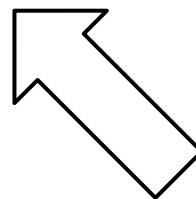
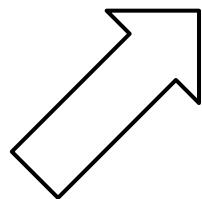
Atenção quanto a serialização dos dados durante a
passivação \ ativação!

**Ei... Como é que eu
obtenho referências para
objetos remotos?**



Serviços de Localização

API

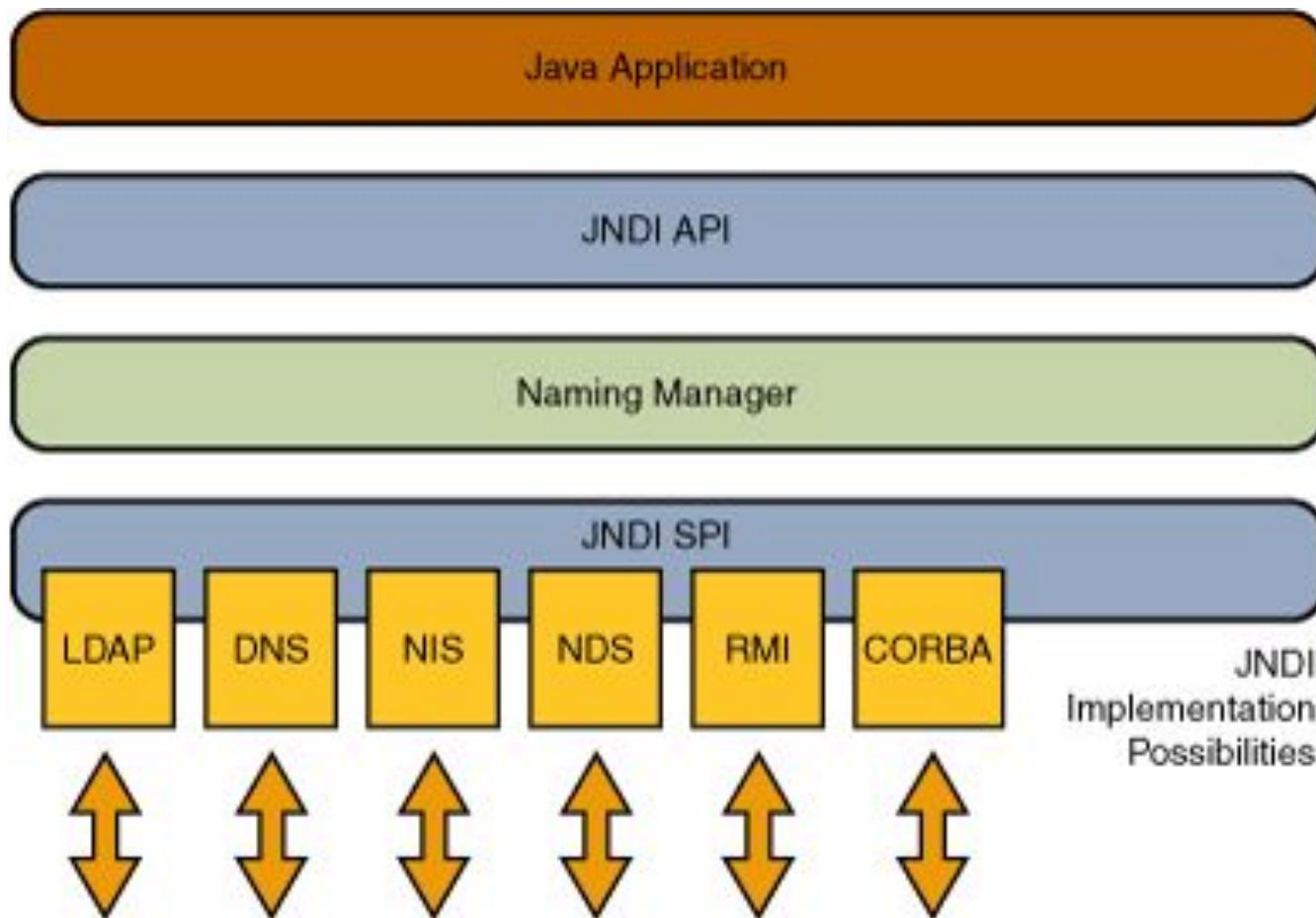


SPI 1

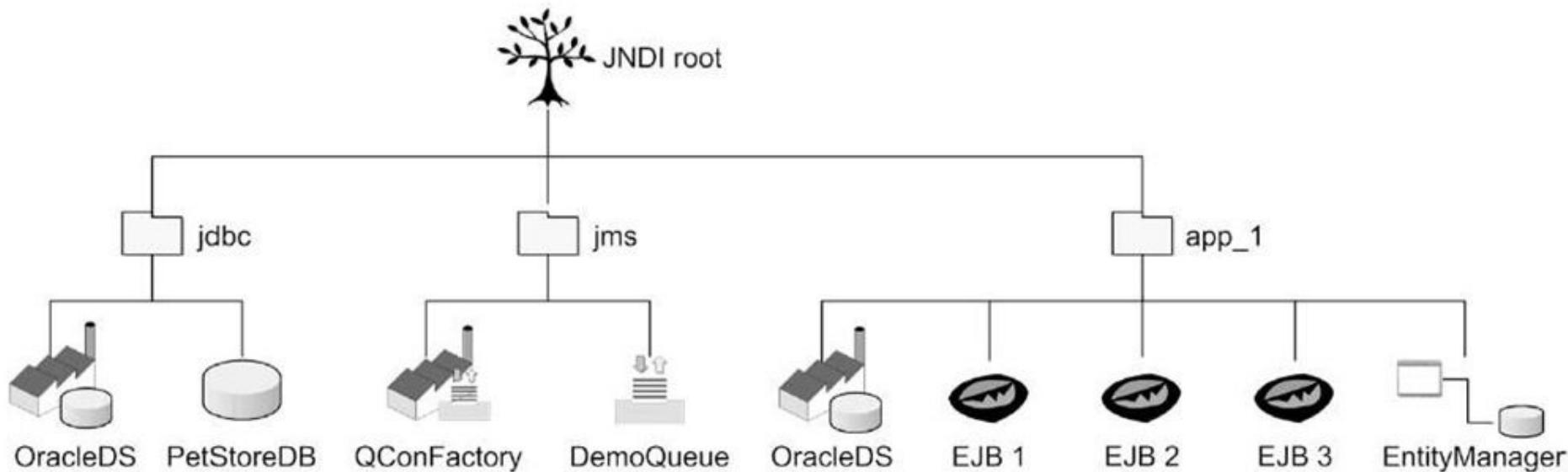


SPI 1

JNDI

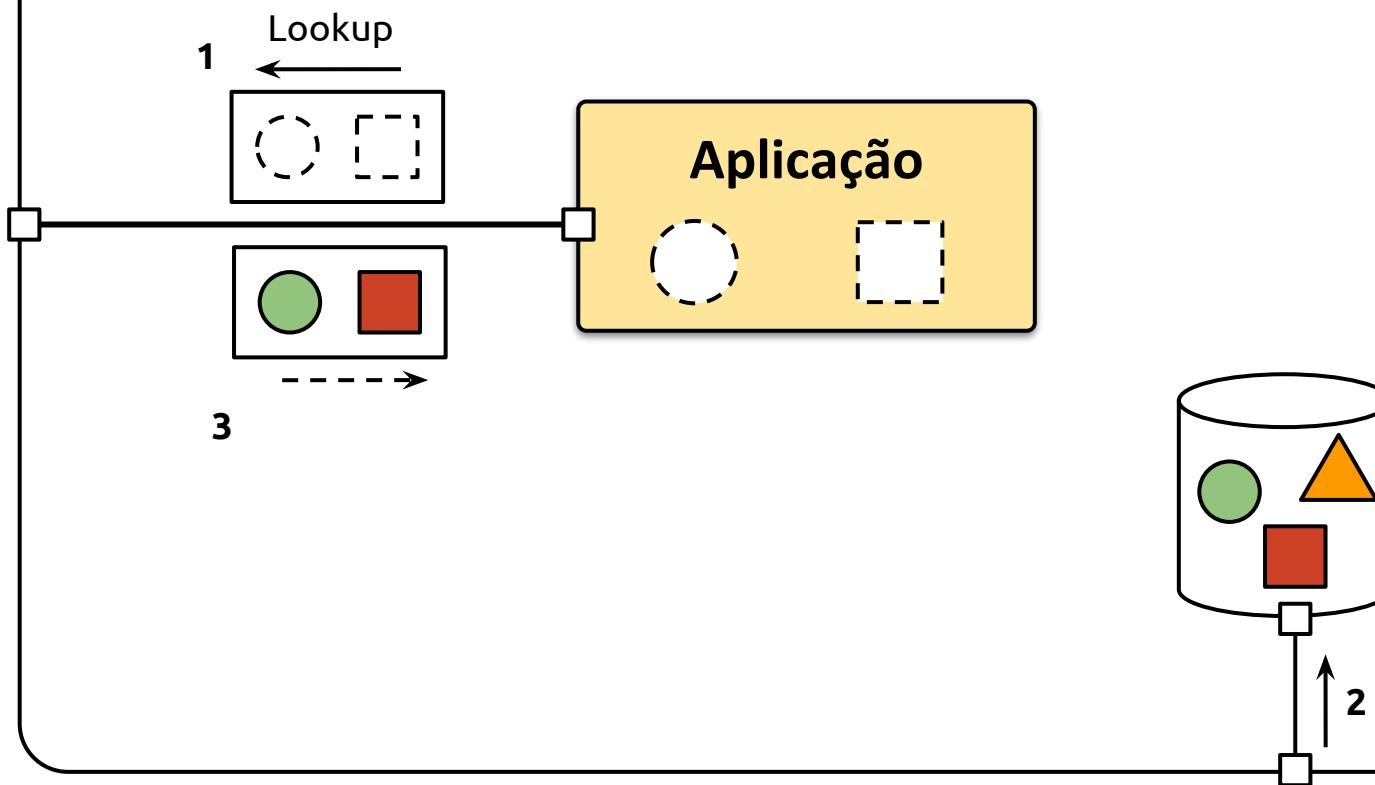


JNDI

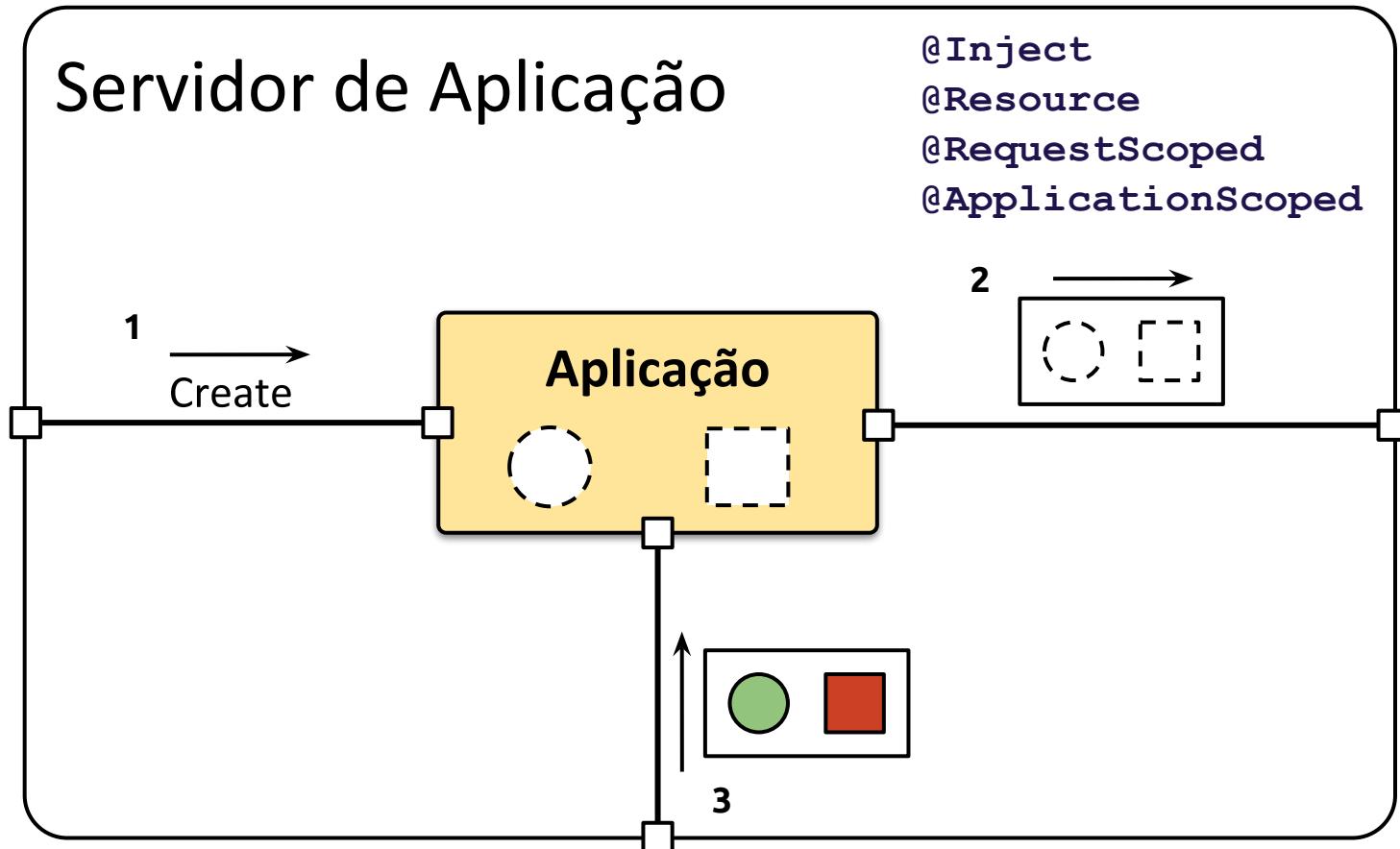


Relembrando...

Servidor de Aplicação



Relembrando...



java:[namespace]/[app]/[modulo]/[bean][!interface]

Namespace	Detalhe
Global	Compartilhado por todos os módulos e componentes do ambiente
App	Nomes compartilhados dentro da mesma aplicação (EAR)
Module	Nomes compartilhados dentro do mesmo módulo (EJBs e WARs)

App: MyApp Módulo: MyModule

```
package uni7;  
...  
@Stateful  
public class MyBean { ... }
```

java:global/MyApp/MyModule/MyBean
java:global/MyApp/MyModule/MyBean!uni7.MyBean
java:app/MyModule/MyBean
java:app/MyModule/MyBean!uni7.MyBean
java:module/MyBean
java:module/MyBean!uni7.MyBean

App: MyApp Módulo: MyModule

```
package uni7;  
...  
@Remote(RemoteInterface.class)  
@Stateless(name="RemoteBean")  
public class MyBean implements RemoteInterface { ... }
```

java:global/MyApp/MyModule/RemoteBean
java:global/MyApp/MyModule/RemoteBean!uni7.RemoteInterface
java:app/MyModule/RemoteBean
java:app/MyModule/RemoteBean!uni7.RemoteInterface
java:module/RemoteBean
java:module/RemoteBean!uni7.RemoteInterface

App: MyApp Módulo: MyModule

```
package uni7;  
...  
@Remote(RemoteInterface.class)  
@Stateless(name="RemoteBean")  
public class MyBean implements RemoteInterface { ... }
```

```
@WebServlet("/MyServlet")  
public class MyServlet extends HttpServlet {  
  
    @EJB(lookup="java:global/MyApp/MyModule/RemoteBean")  
        RemoteInterface bean;  
}
```

Exercício 02



Exercício 02

Crie e implante um exemplo de Stateless e Stateful Session Bean. Anote métodos para registrar seus ciclos de vida no console do servidor de aplicação. Após isso, cheque seus nomes no serviço JNDI do servidor.

Dicas

- Utilizaremos um projeto multi-módulo
- Plugins Maven
 - maven-war-plugin
 -

Crie o projeto pai

```
$ mvn -B archetype:generate  
-DarchetypeGroupId=org.codehaus.mojo.archetypes  
-DarchetypeArtifactId=pom-root  
-DarchetypeVersion=RELEASE -DgroupId=uni7  
-DartifactId=project
```

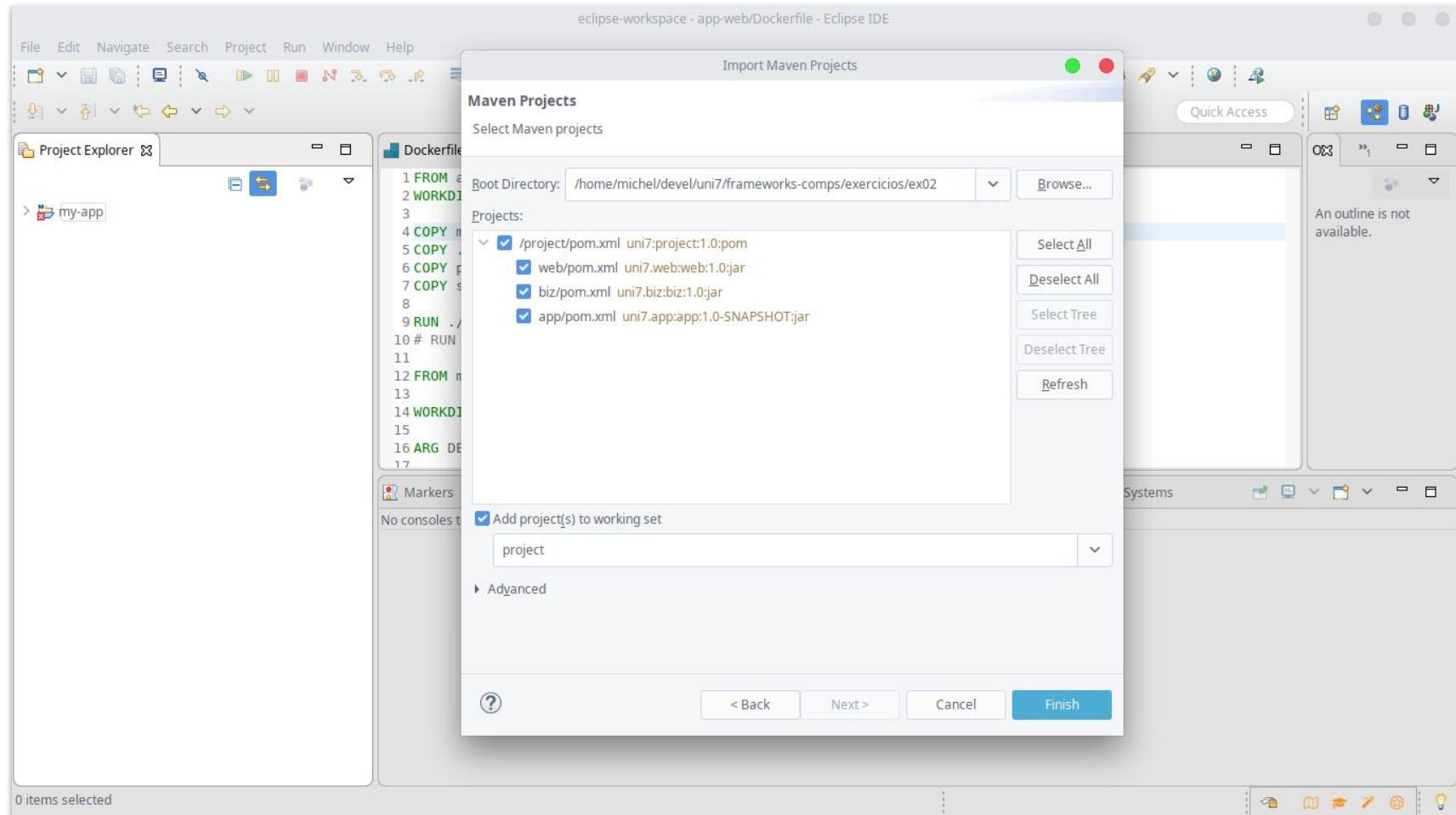
Crie os módulos

```
mvn -B archetype:generate  
-DarchetypeGroupId=org.apache.maven.archetypes  
-DgroupId=uni7.app -DartifactId=app
```

```
$ mvn -B archetype:generate  
-DarchetypeGroupId=org.apache.maven.archetypes  
-DgroupId=uni7.web -DartifactId=web
```

```
mvn -B archetype:generate  
-DarchetypeGroupId=org.apache.maven.archetypes  
-DgroupId=uni7.biz -DartifactId=biz
```

Importe os projetos



Configure os arquivos pom.xml

- Ajuste o empacotamento de cada projeto
 - war, ejb e ear
- Inclua e configure os plugins necessários
 - maven-war-plugin
 - maven-ejb-plugin
 - maven-ear-plugin
- Ajuste as dependências no projeto da aplicação

Configure os arquivos pom.xml

- Inclua as dependências da API Java EE em cada projeto
 - javaee-api
 - javaee-web-api

LET'S HAVE SOME



Trabalho 01



Trabalho 01

<https://github.com/michelav-uni7/loja-virtual>