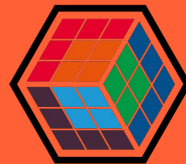


# Testes de contrato para microserviços orientados a eventos utilizando o Pact



# Olá!



Ana Ludmila  
QA - Thoughtworks



Bruna Gonçalves  
QA - Thoughtworks

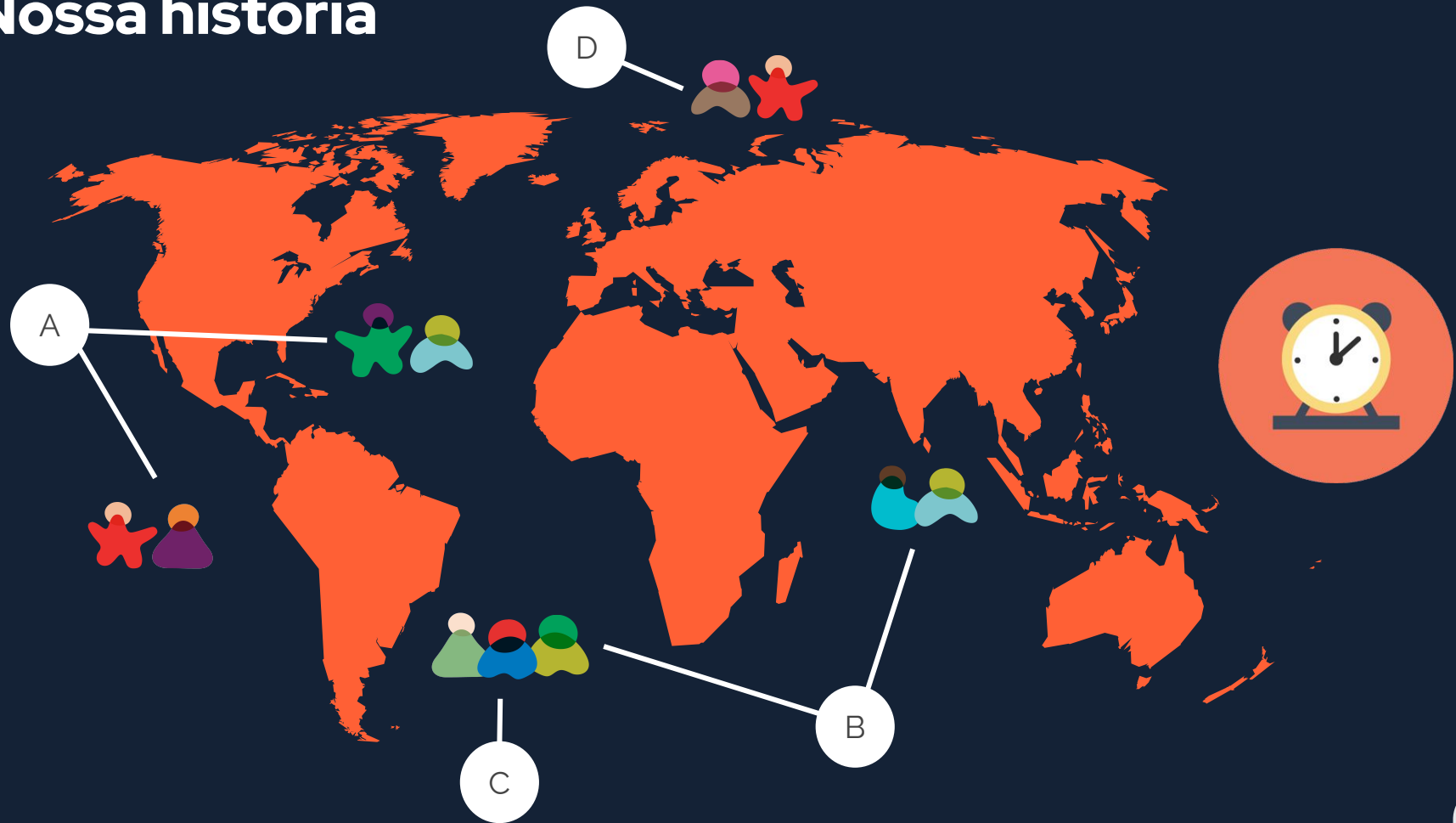
# Expectativas

- A nossa história
- E qual foi o problema?
- Pausa para a pirâmide de testes
- Testes de contrato
- CDC vs PDC
- Detalhar a abordagem utilizada - CDC
- Exemplos

## A nossa história

- Sistema em desenvolvimento distribuído;
- Equipes diferentes trabalhando para uma mesma entrega;
- Prazos diversos para entrega de cada parte do sistema;
- Tempo curto até a entrega! 🕒

# Nossa história



O problema



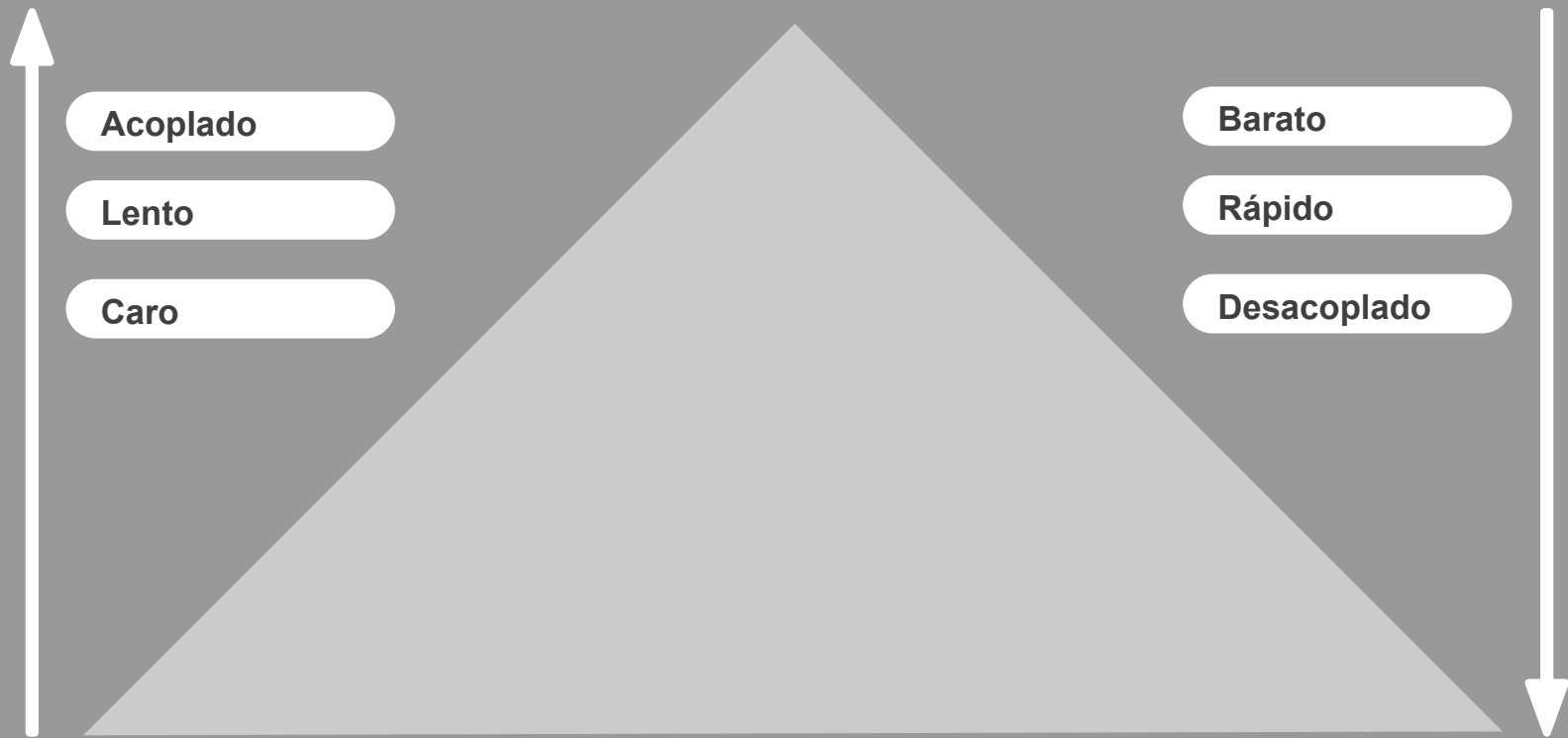


1

# Pirâmide de Testes

Pausa para a Pirâmide de Testes





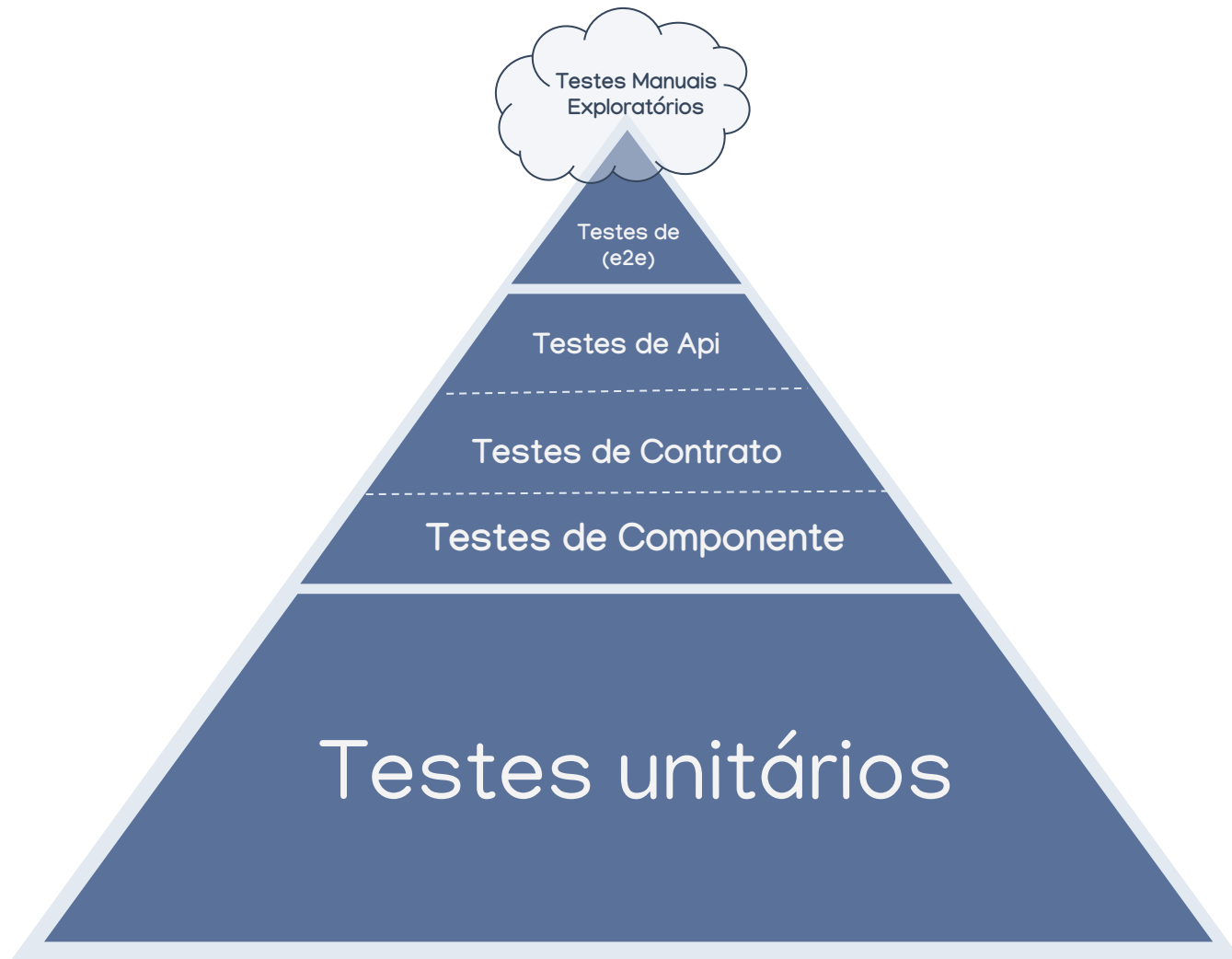




**E2E**

**Integração**

**Unitários**





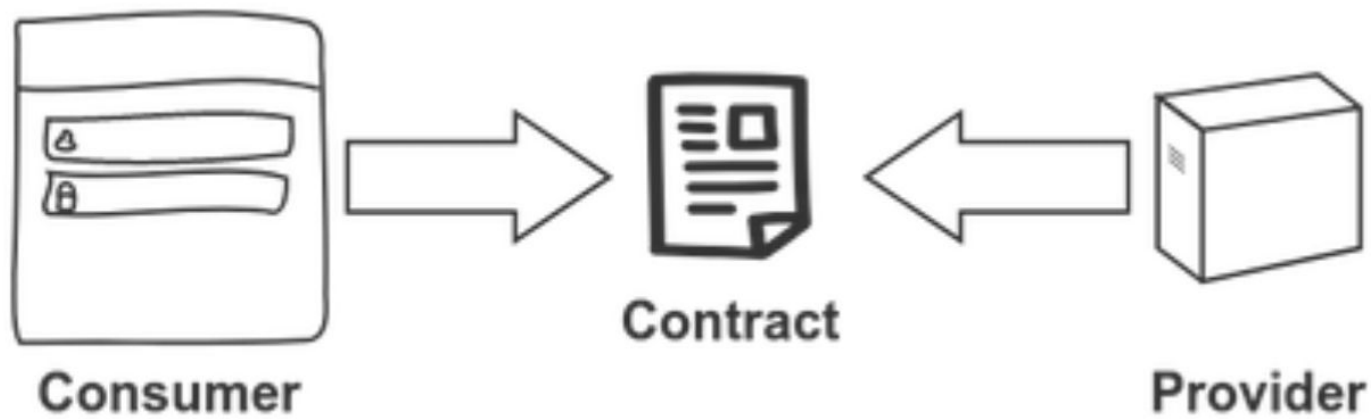
2

# Testes de Contrato

Uma breve introdução



Tem como objetivo garantir que o conteúdo fornecido não tenha sido modificado, podemos dizer que tem a finalidade de **validar se o contrato acordado foi ou não quebrado**; deve validar se o schema permanece o mesmo garantindo assim a integridade dos dados na comunicação entre cliente/servidor.





3

# Abordagens

## Consumer-Driven Contract

Os Consumidores definem suas expectativas e o Provedor verifica se elas foram cumpridas

## Provider-Driven Contract

O Provedor define o contrato e os Consumidores verificam se são compatíveis com ele



**Revela interfaces não utilizadas**



**Revela atributos de interfaces não utilizados**

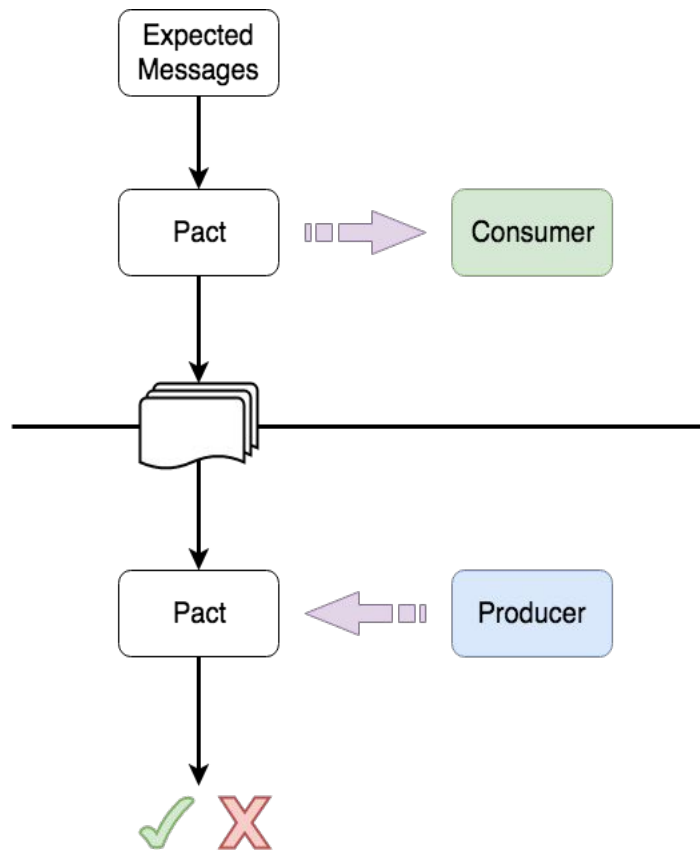




# CDC

Permite que os consumidores dirijam a implementação de um contrato.

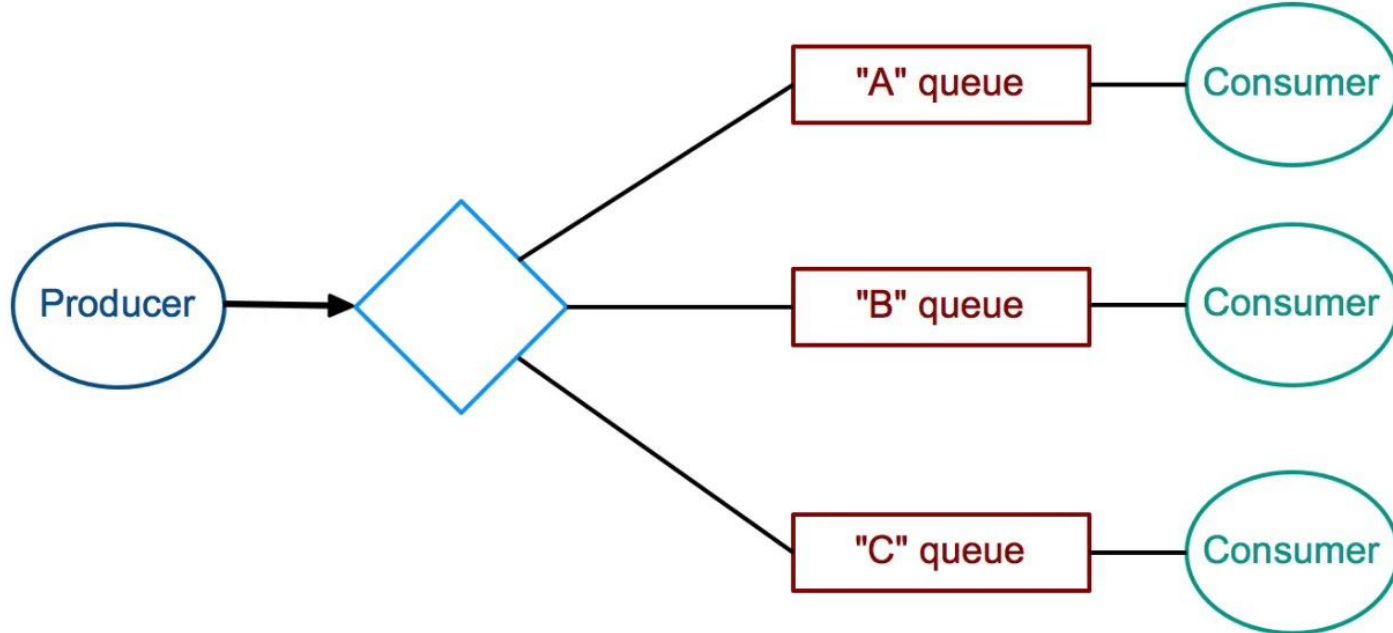
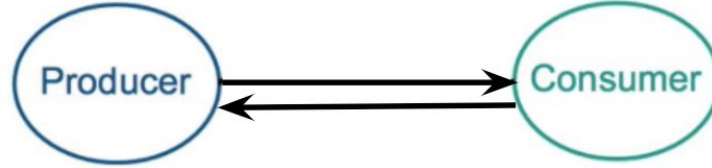
Usando o *CDC*, os consumidores de uma interface **geram contratos** e **executam testes que verificam as interfaces** consumidas. A equipe consumidora **pública esses contratos** para que a equipe provedora possa **buscar e executar esses testes** facilmente. A equipe provedora agora pode desenvolver sua aplicação executando os testes do *CDC*. Depois que todos os testes passam, eles sabem que implementaram tudo que a equipe consumidora precisa.





4

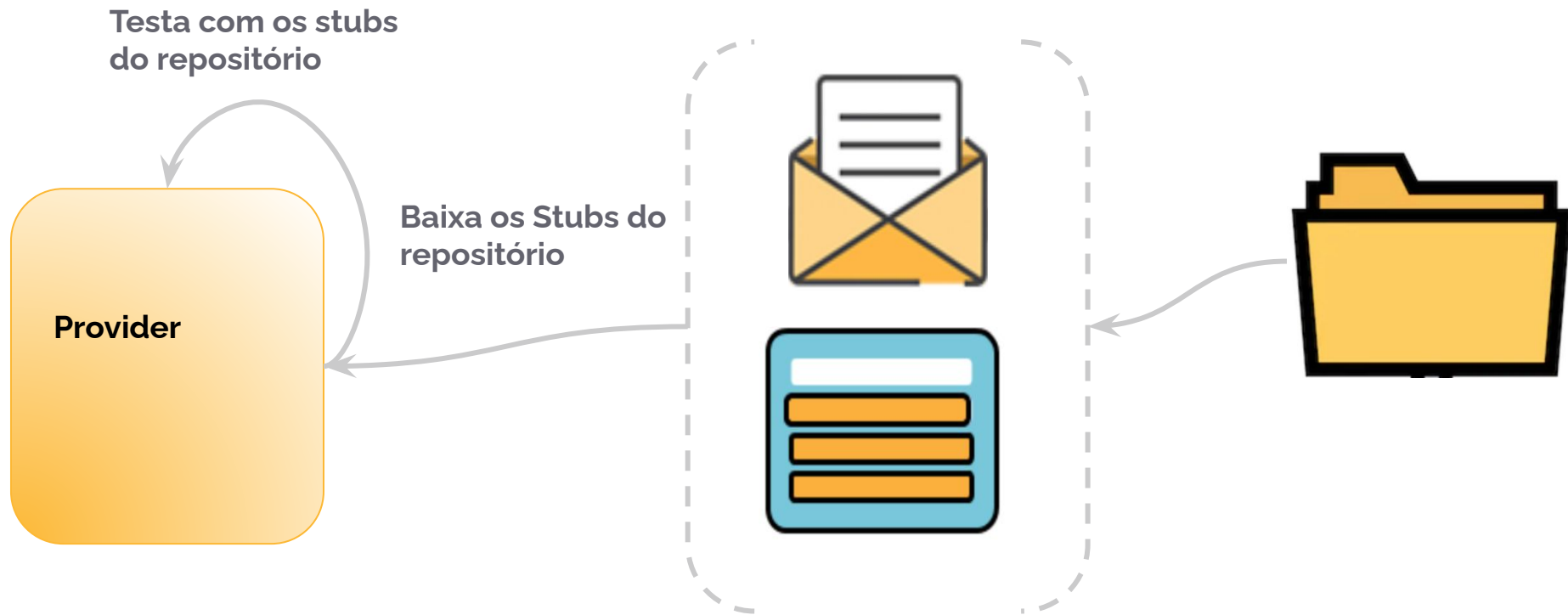
# HTTP vs AMQP

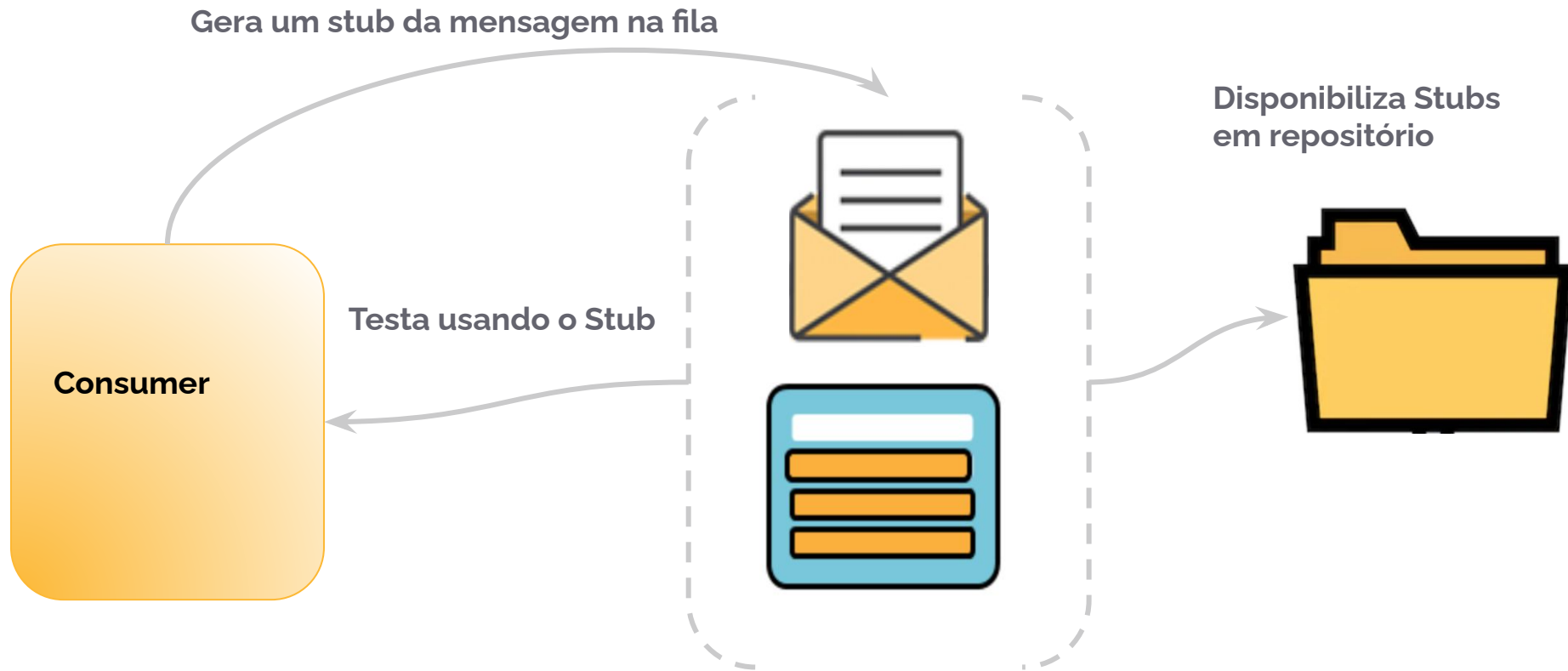




5

**Exemplo**





# Referências

- <https://martinfowler.com/articles/practical-test-pyramid.html>
- <https://docs.pact.io/>
- <https://medium.com/john-lewis-software-engineering/consumer-driven-contract-testing-a-scalable-testing-strategy-for-microservices-3f2b09f99ed1>
- <https://reflectoring.io/cdc-pact-messages/>
- <https://blog.codecentric.de/en/2019/11/message-pact-contract-testing-in-event-driven-applications/>
- [Repositório Consumer](#)
- [Repositório Provider](#)



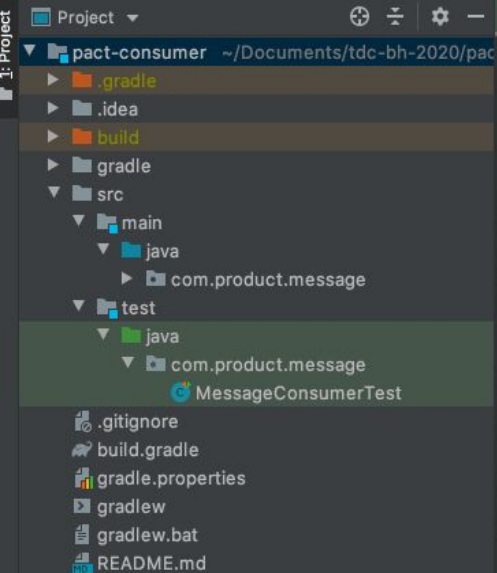
# Obrigada!



Ana Ludmila  
QA - Thoughtworks



Bruna Gonçalves  
QA - Thoughtworks

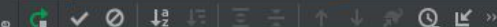


```

1 package com.product.message;
2
3 import ...
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18 @RunWith(SpringRunner.class)
19 @SpringBootTest
20 public class MessageConsumerTest {
21
22     @Rule
23     public MessagePactProviderRule mockProvider = new MessagePactProviderRule( testClassInstance: this);
24     private byte[] currentMessage;
25
26     @Autowired
27     private MessageConsumer messageConsumer;
28
29     @Pact(provider = "productservice", consumer = "productclient")
30     @Pact
31     public MessagePact productCreatedMessagePact(MessagePactBuilder builder) {
32         PactDslJsonBody body = new PactDslJsonBody();
33         body.stringType("messageUuid");
34         body.object("product")

```

Run:  MessageConsumerTest 



▼ \* Instantiating tests...

```
> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes
> Task :compileTestJava
> Task :processTestResources NO-SOURCE
> Task :testClasses
> Task :test
```

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.reflection.CachedClass (file:/Users/bngoncalv/.gradle/caches/modules-2/files-2.1/c
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy.reflection.CachedClass
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

ProductCreatedMessageProviderTest

ProductCreatedMessageProviderTest

Project

pac

src

test

java

com

product

message

ProductCreatedMessageProviderTest

ProductCreatedMessageProviderTest.java

```
1 package com.product.message;
2
3 import ...
4
5 @RunWith(PactRunner.class)
6 @Provider("productservice")
7 @PactFolder("../pact-consumer/target/pacts")
8 public class ProductCreatedMessageProviderTest {
9
10     @TestTarget
11     public final Target target = new CustomAmpTarget(Collections.singletonList("com.product.message"));
12
13     private MessagePublisher publisher = Mockito.mock(MessagePublisher.class);
14
15     private MessageProvider messageProvider = new MessageProvider(new ObjectMapper(), publisher);
16
17     @PactVerifyProvider("a product created message")
18     public String verifyProductCreatedMessage() throws IOException {
19         // given
20         doNothing().when(publisher).publishMessage(any(String.class), eq(value: "product.created"));
21
22         // when
23         ProductCreatedMessage message = ProductCreatedMessage.builder()
24             .messageId(UUID.randomUUID().toString())
25             .build();
26         return messageProvider.getMessage(message);
27     }
28 }
```

Run: ProductCreatedMessageProviderTest

Instantiating tests...

Gradle Daemon started in 7 s 204 ms

> Task :cleanTest

> Task :compileJava

> Task :processResources UP-TO-DATE

> Task :classes

> Task :compileTestJava

> Task :processTestResources NO-SOURCE

> Task :testClasses

> Task :test