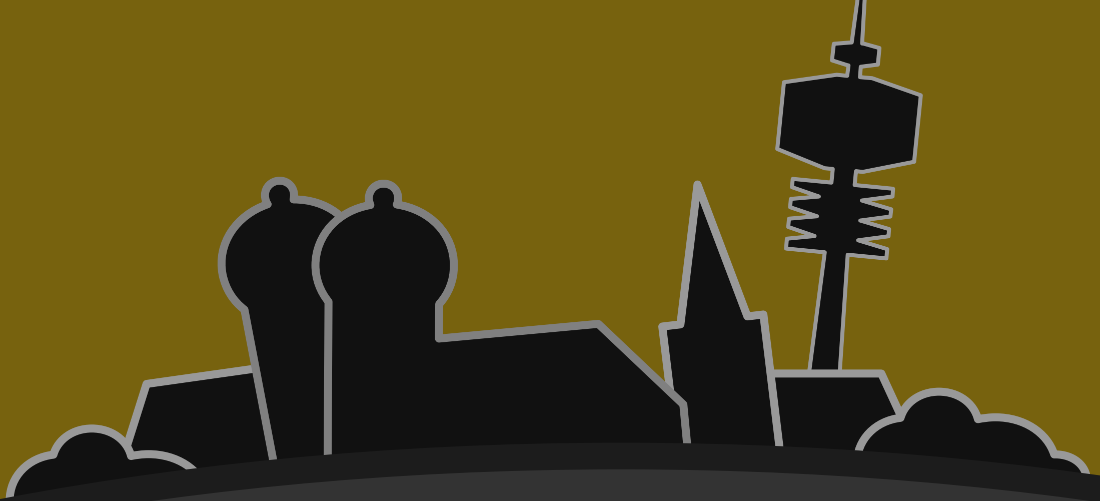
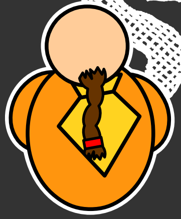
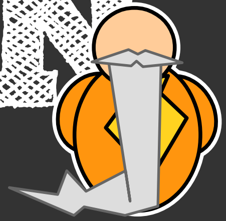


Wenn

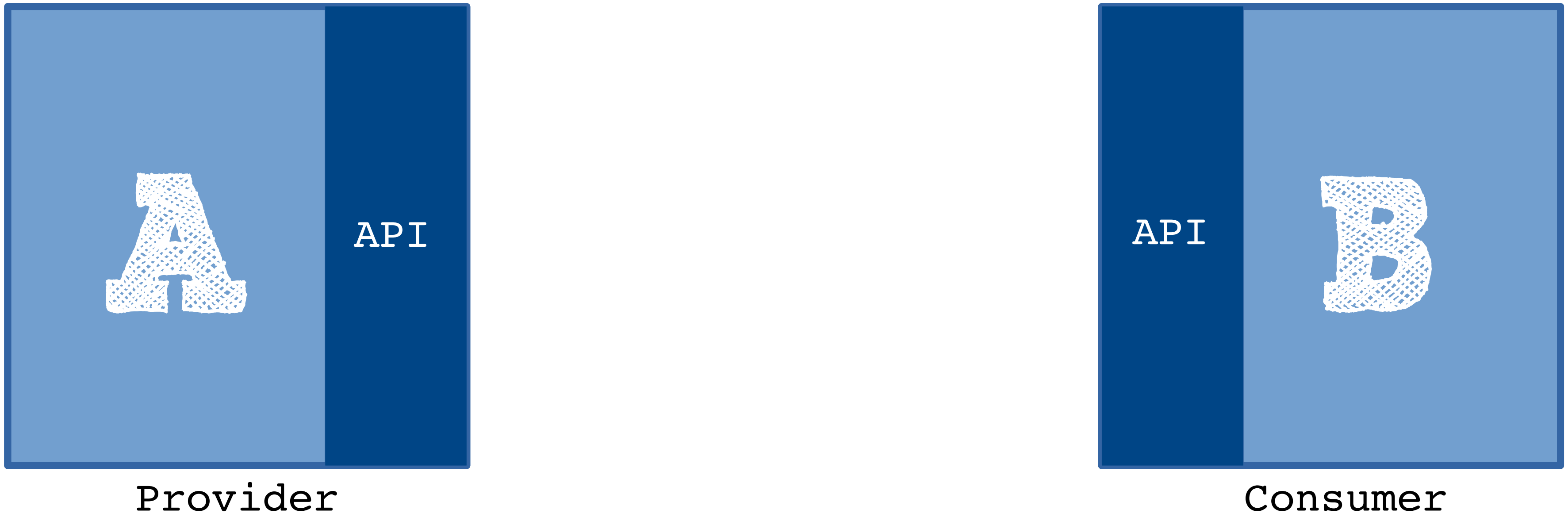
SCHNITTSTELLEN

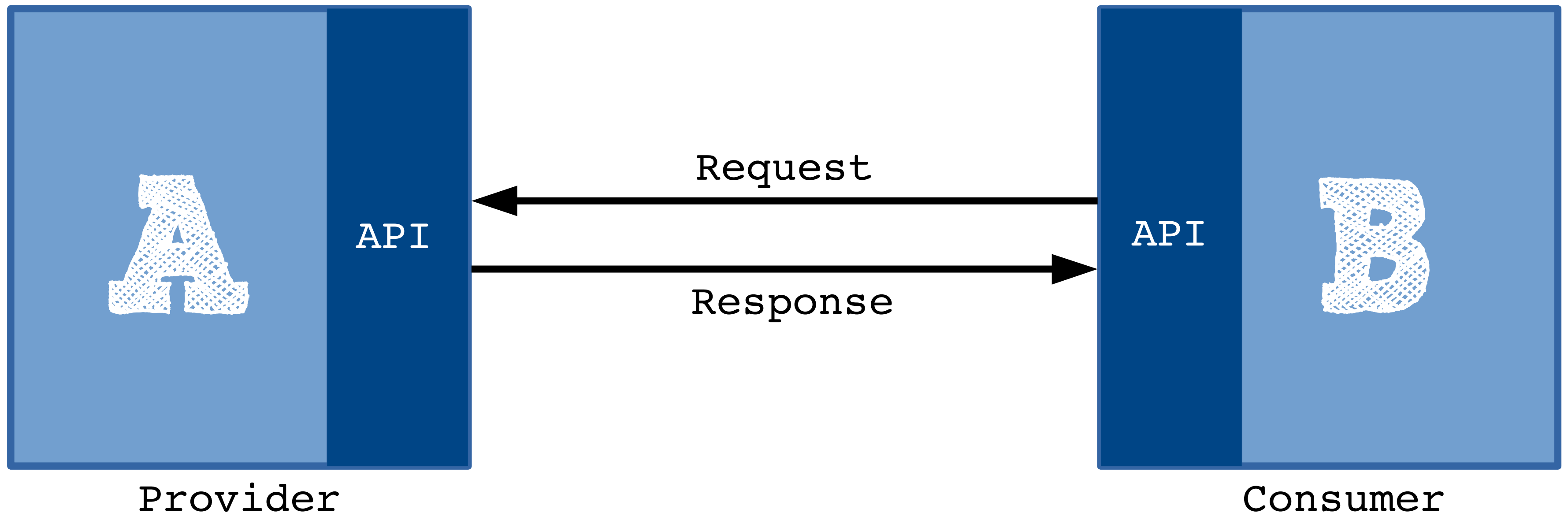


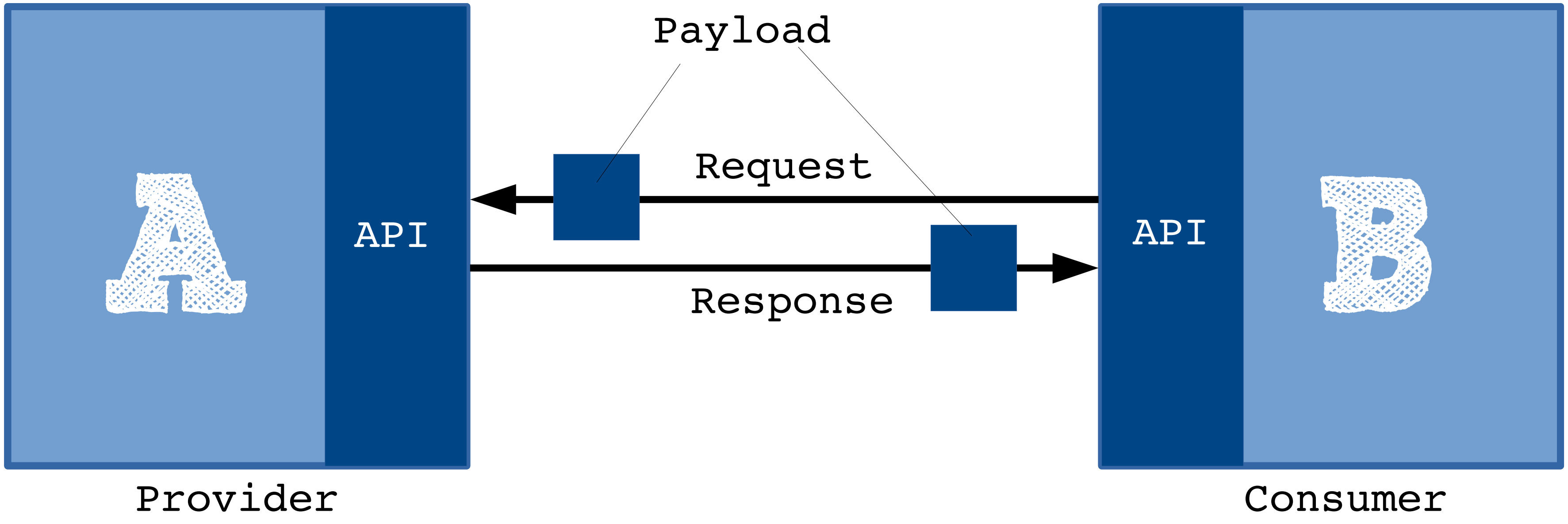


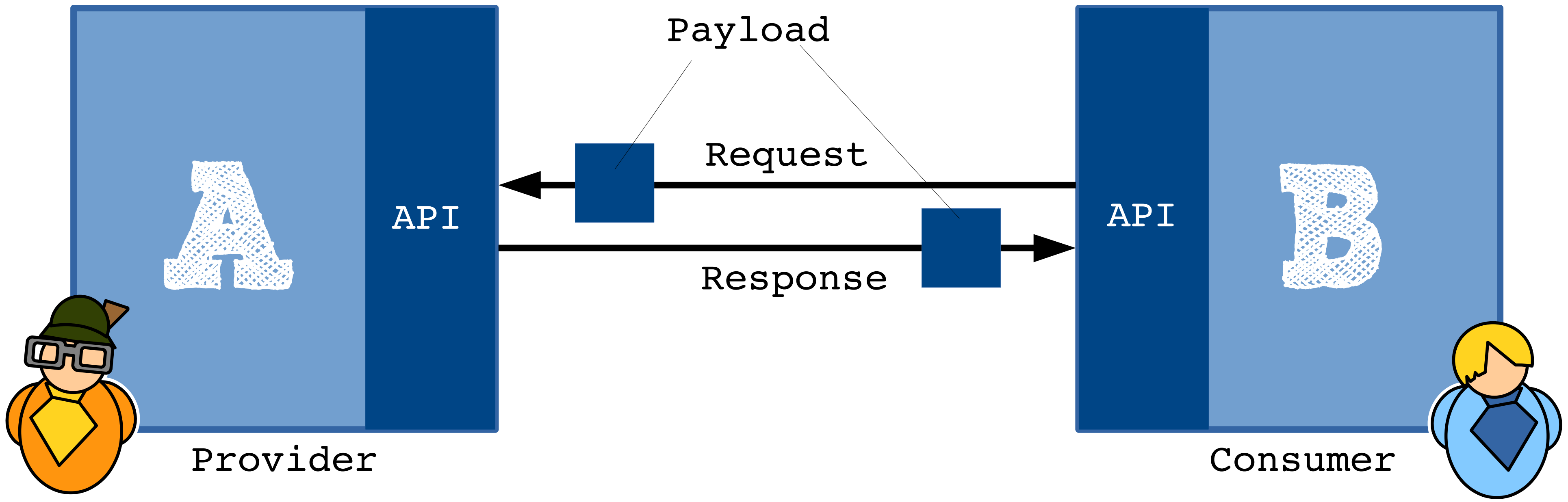
alt werden...

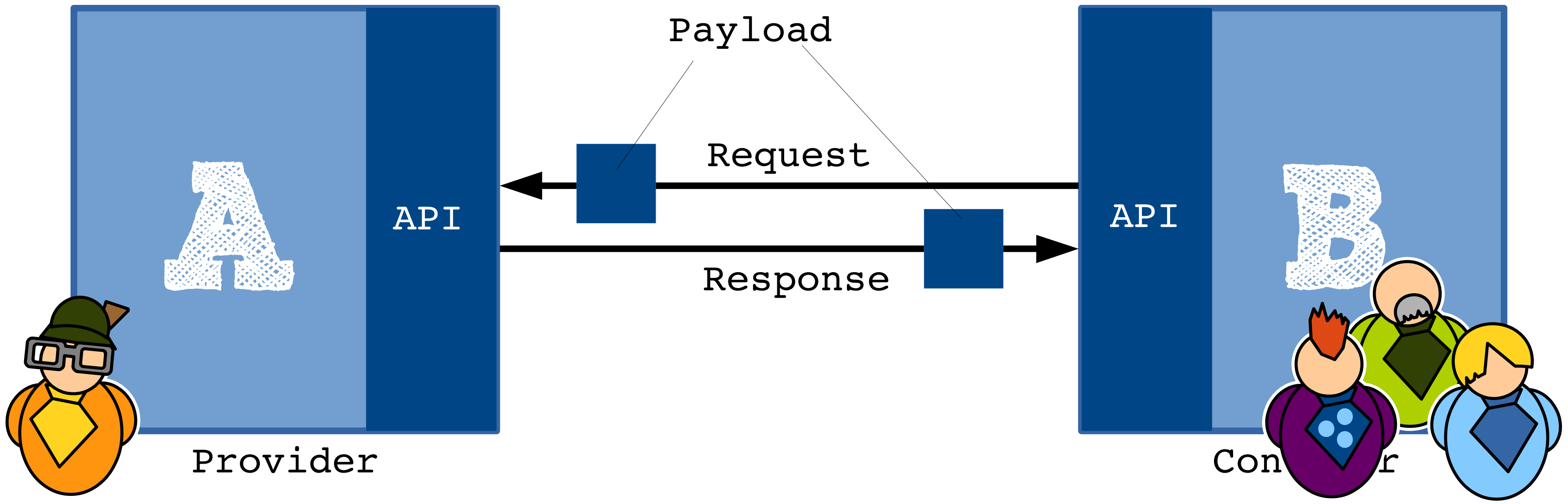
claus.straube@muenchen.de

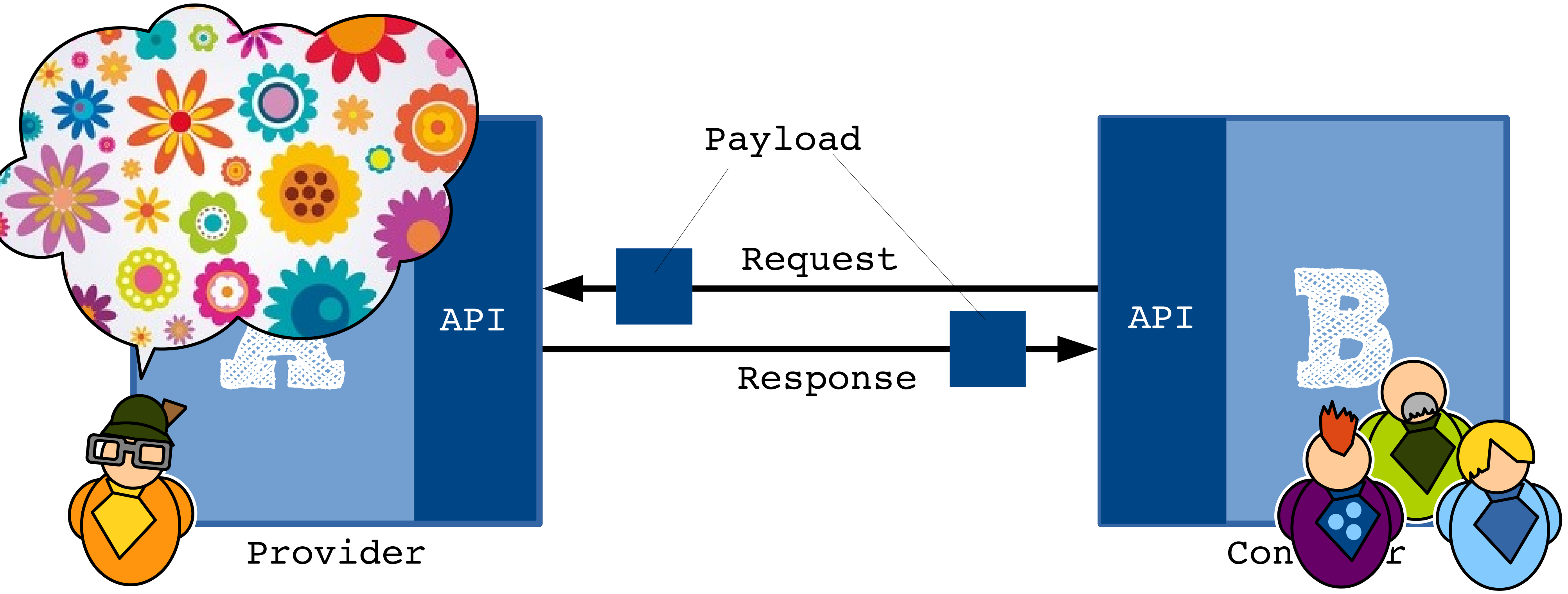

A
API
Provider
API
B
Consumer

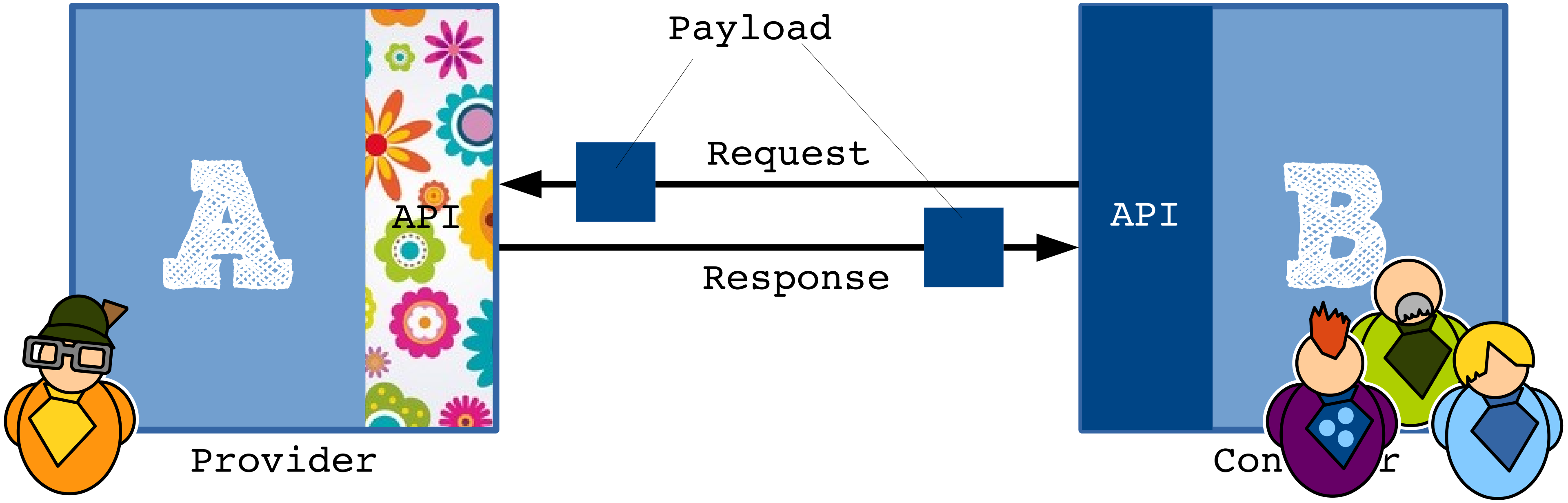

A
API
Request
Response
API
B
Provider
Consumer

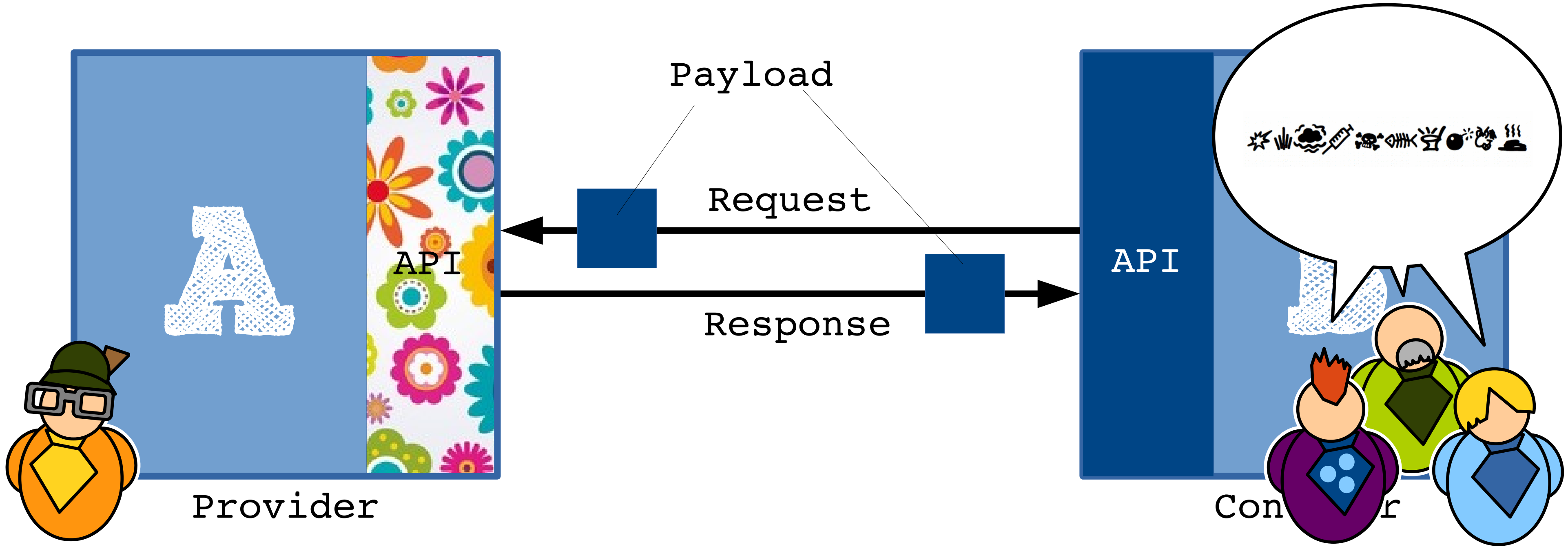

Payload
Request
Response
A
API
API
B
Provider
Consumer

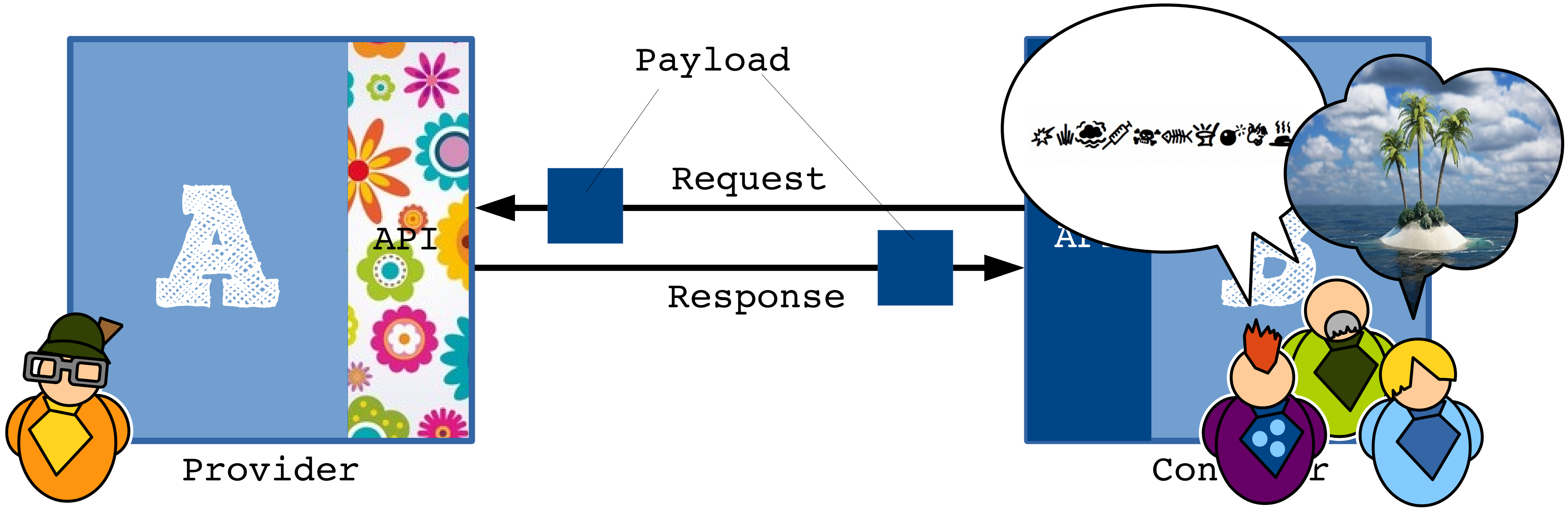

Payload
Request
Response
A
API
API
B
Provider
Consumer

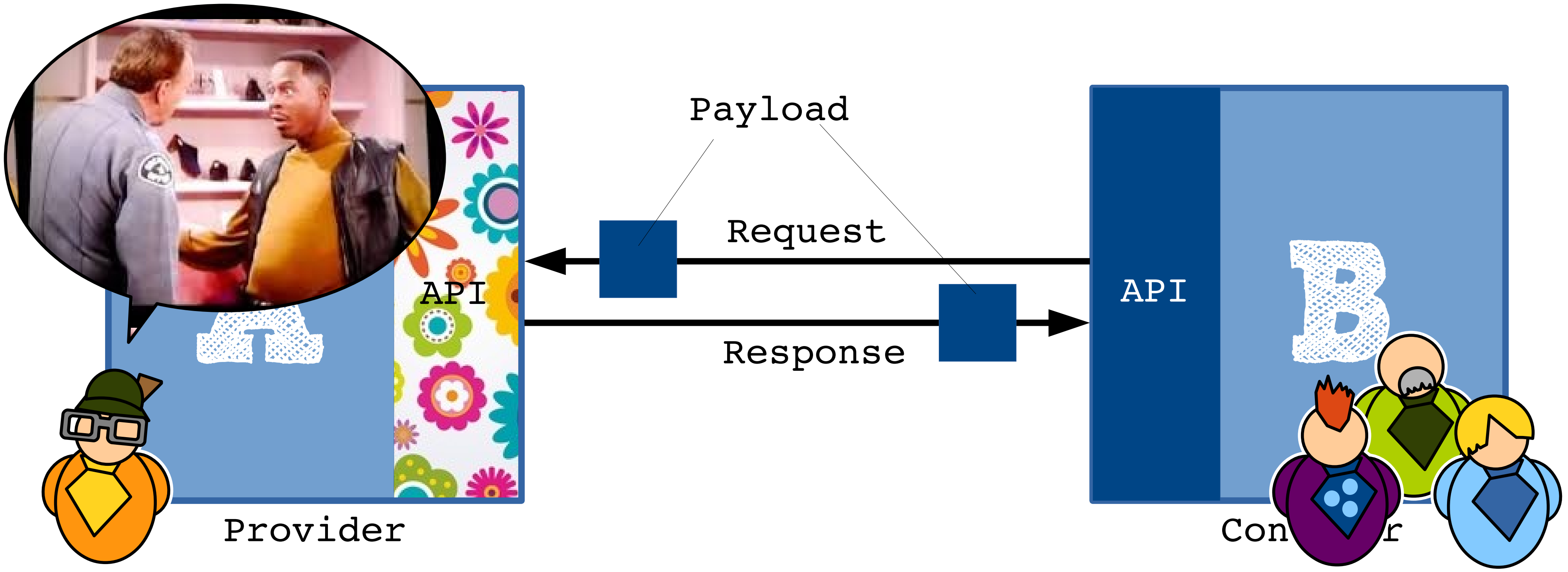

Payload
Request
Response
A
API
API
B
Provider
Con r

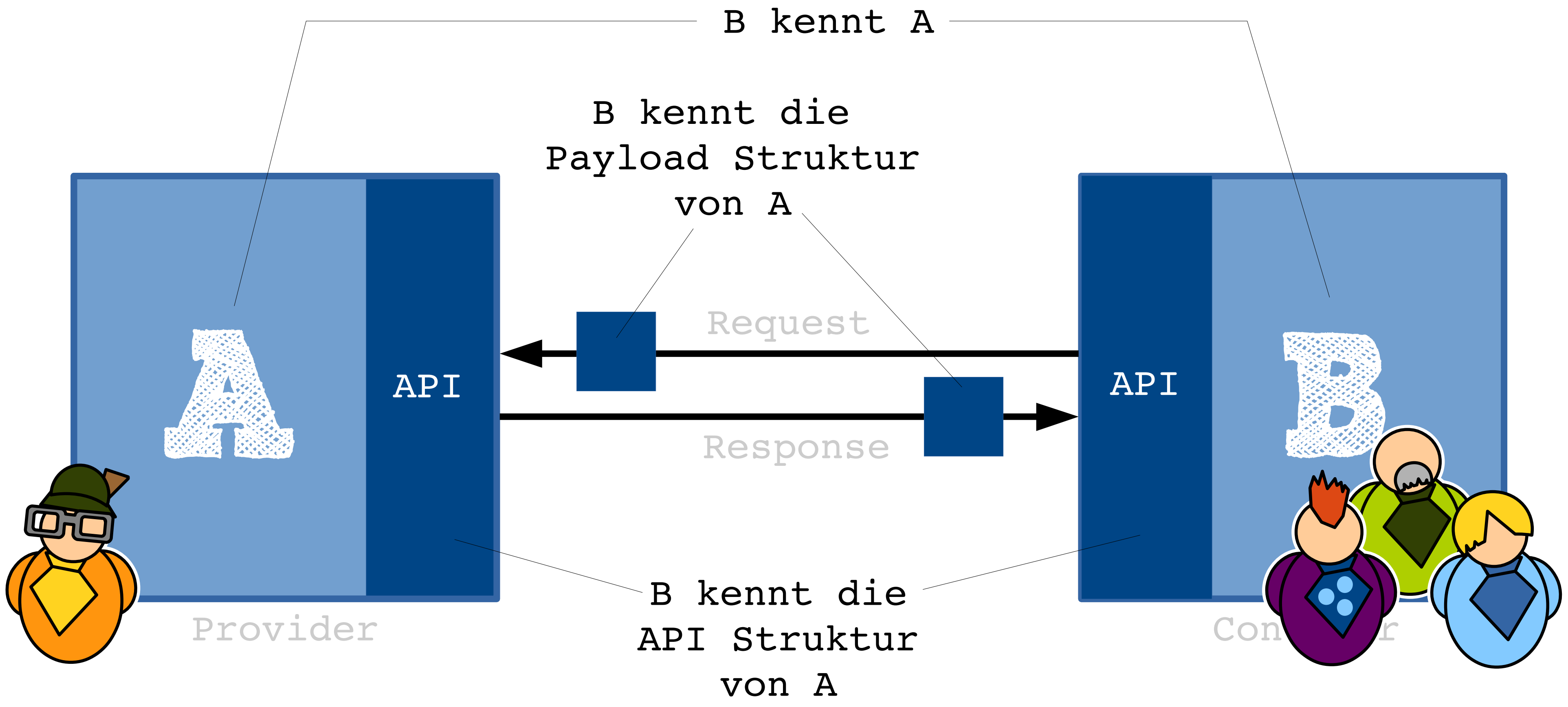

Payload
Request
Response
API
API
A
B
Provider
Con r


Payload
Request
Response
A
API
B
API
Provider
Con r


Payload
Request
Response
API
API
Provider
Con r

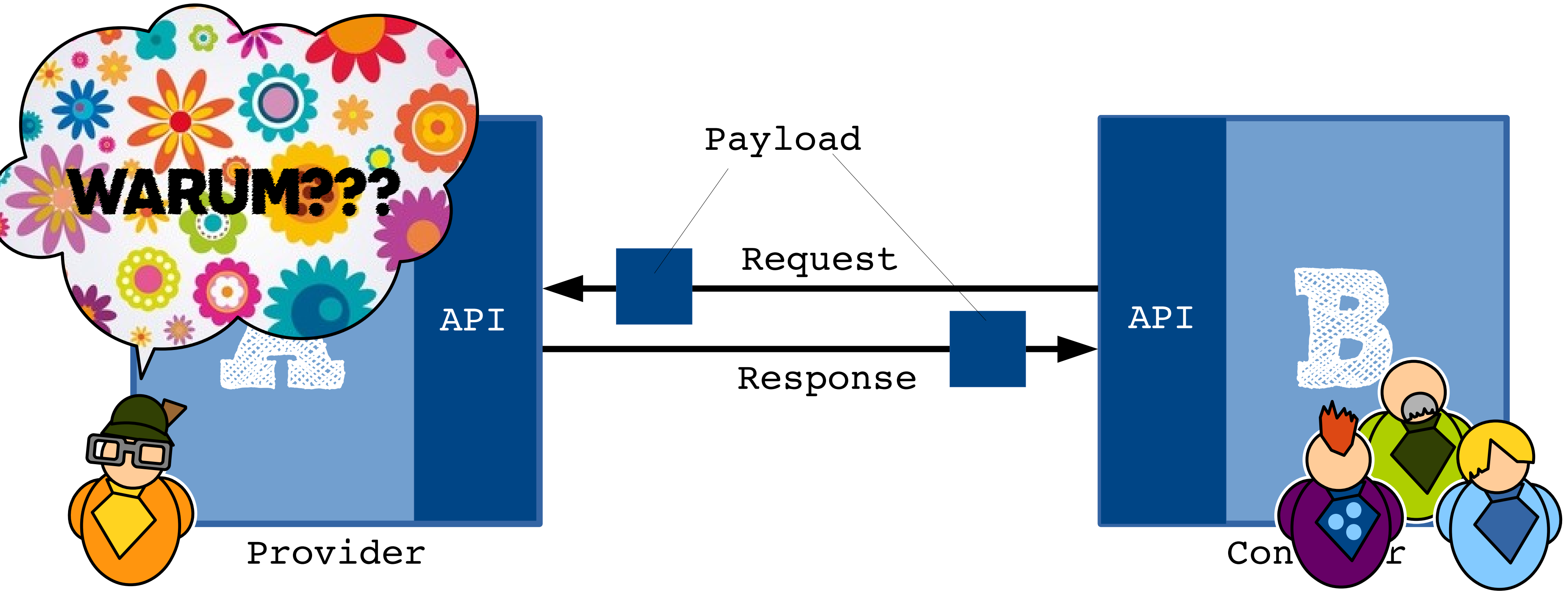

Payload
Request
Response
A
API
Provider
API
B
Consumer

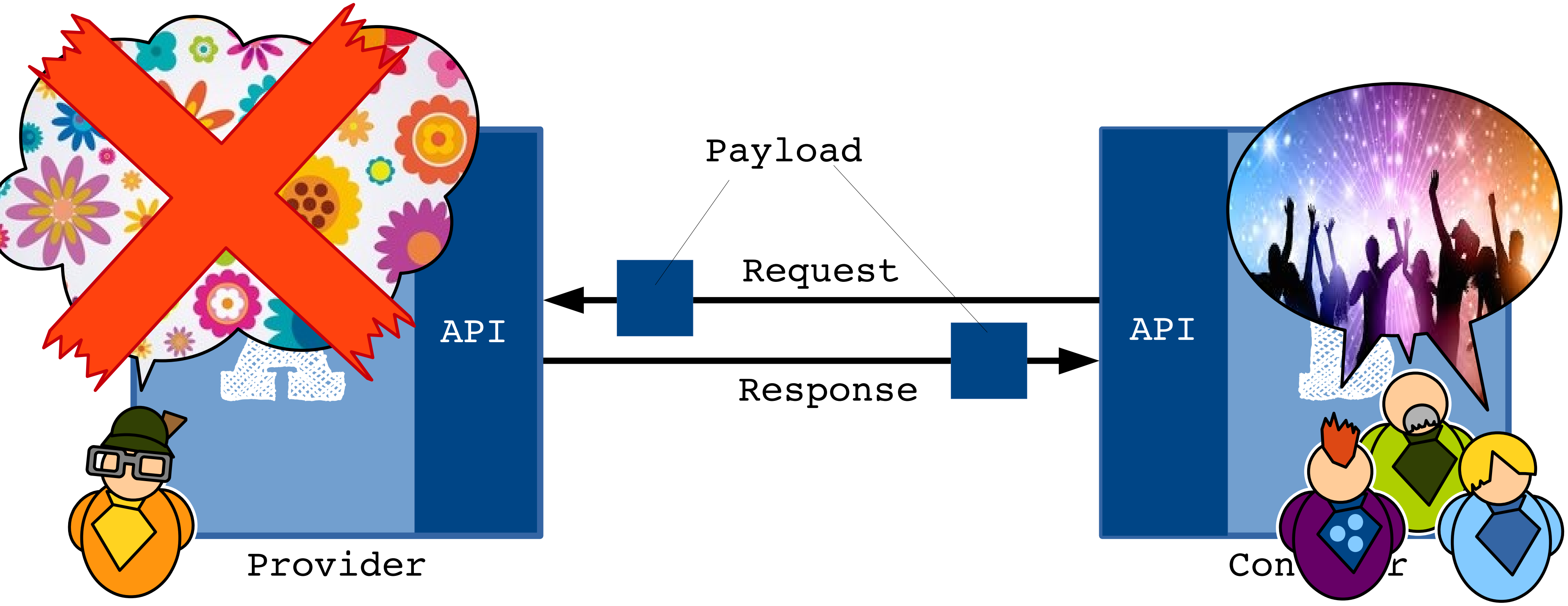

Payload
Request
Response
API
API
B
Provider
Con r


B kennt A
B kennt die Payload Struktur von A
Request
Response
A
API
API
B
Provider
Con
B kennt die API Struktur von A

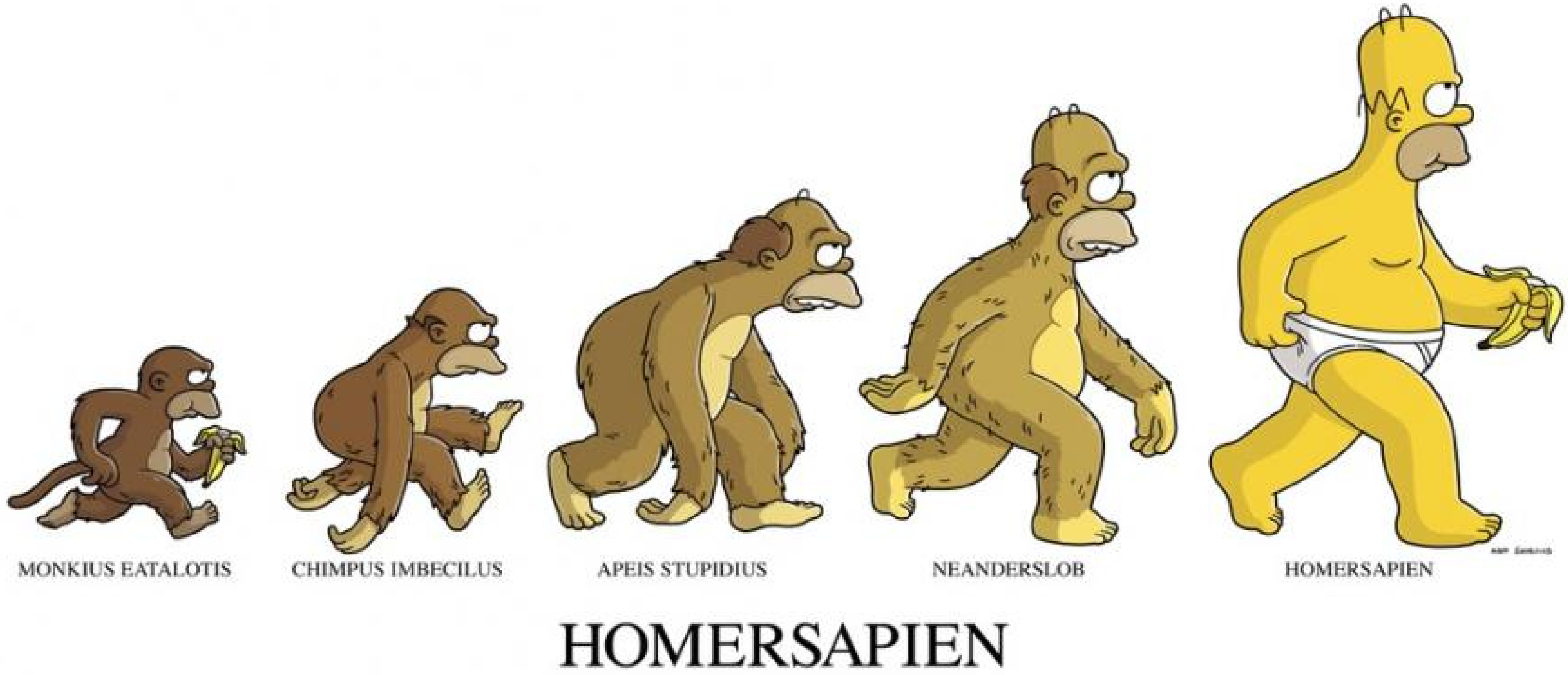
System B muss ziemlich viel von System

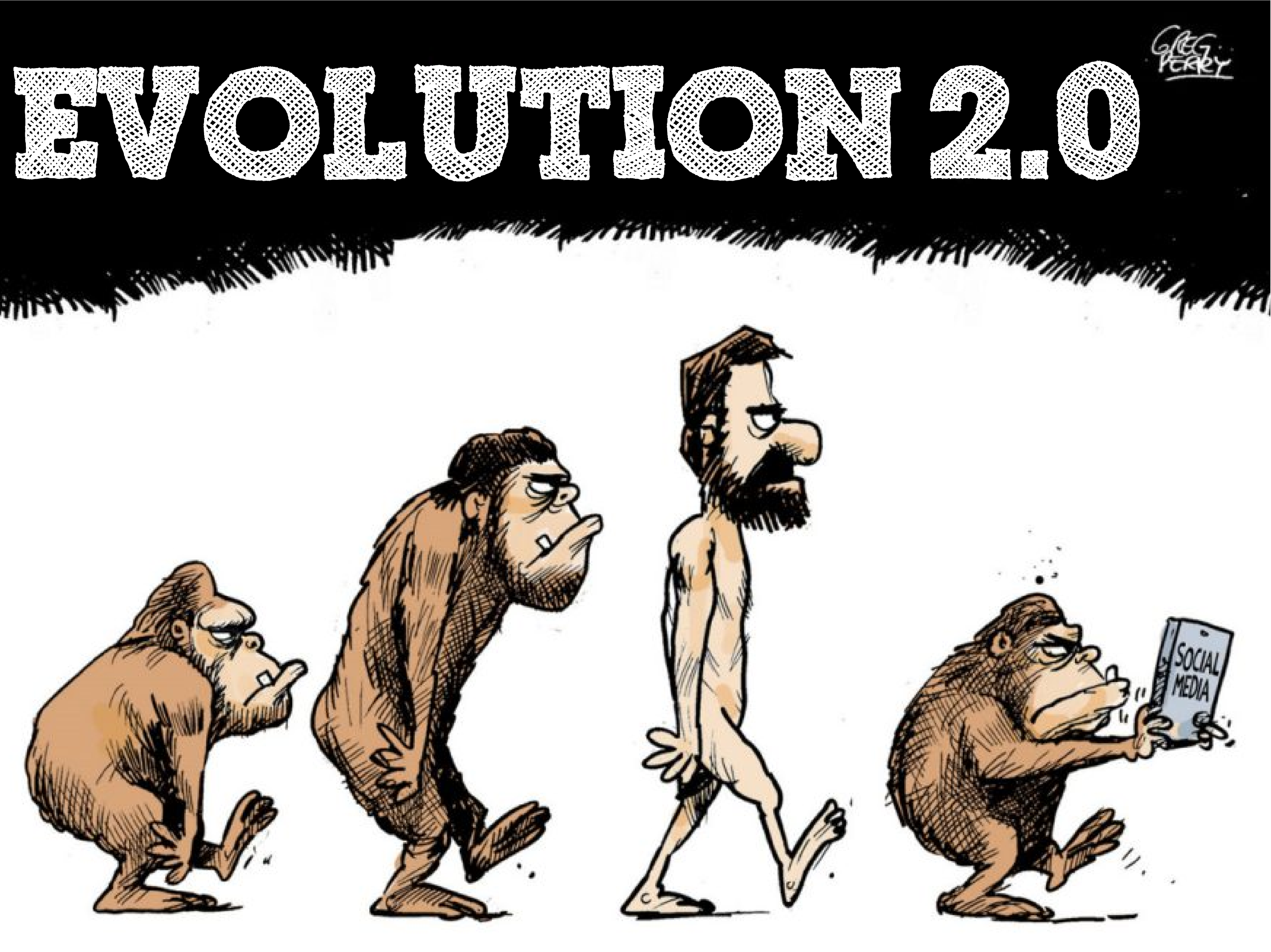
A kennen, damit die Kommunikation funktioniert!

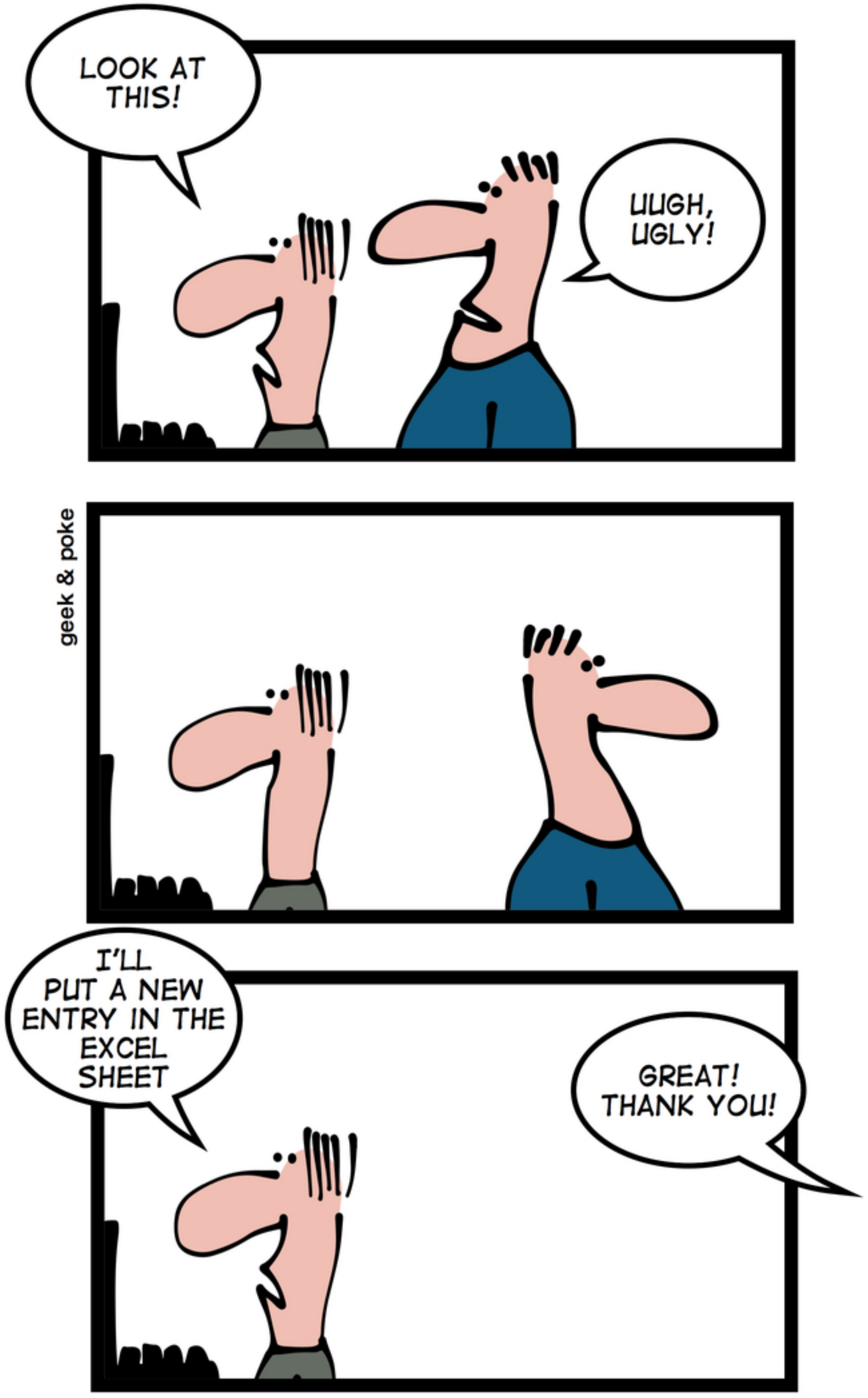

WARUM???
API
Provider
Payload
Request
Response
API
B
Con r


Payload
Request
Response
API
API
Provider
Con r

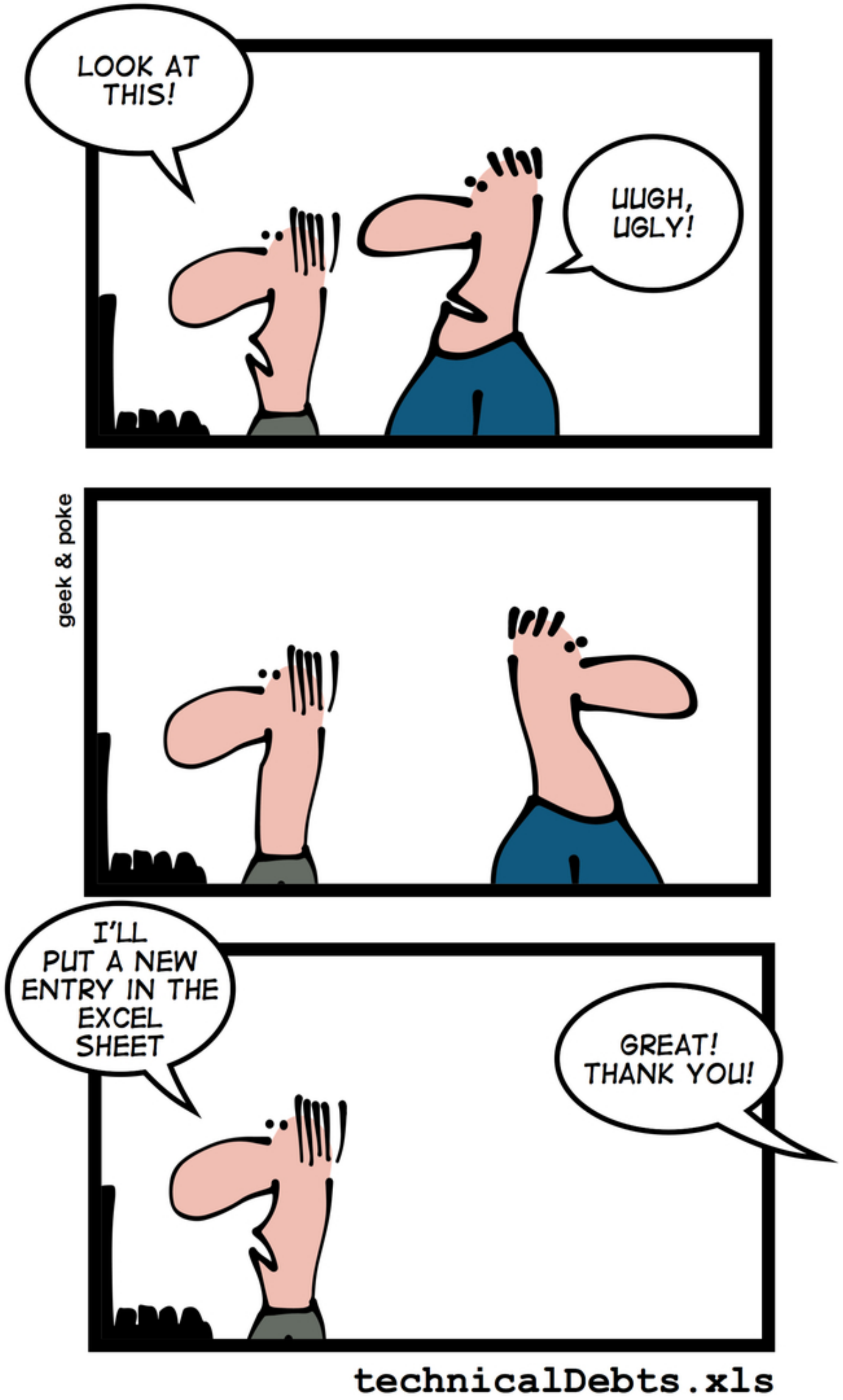
EVOLUTION


MONKIUS EATALOTIS
CHIMPUS IMBECILUS
APEIS STUPIDIUS
NEANDERSLOB
HOMERSAPIEN
HOMERSAPIEN

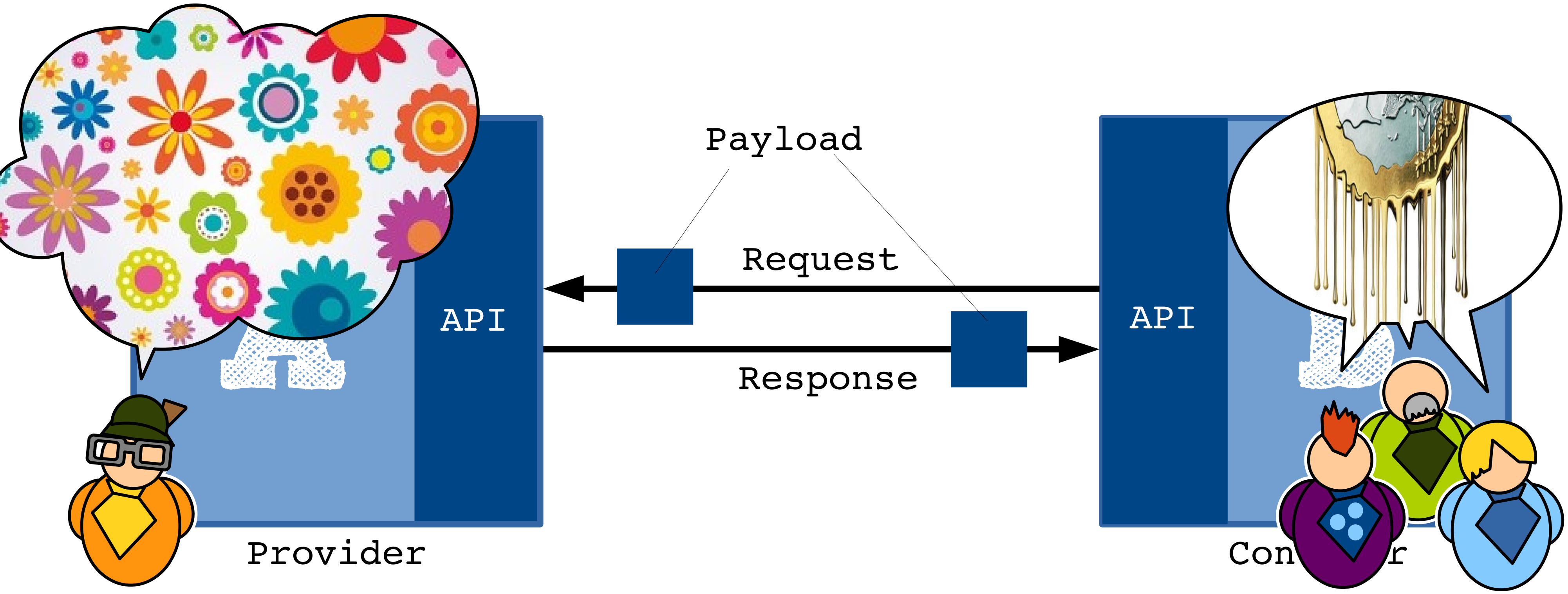

EVOLUTION 2.0
SOCIAL MEDIA

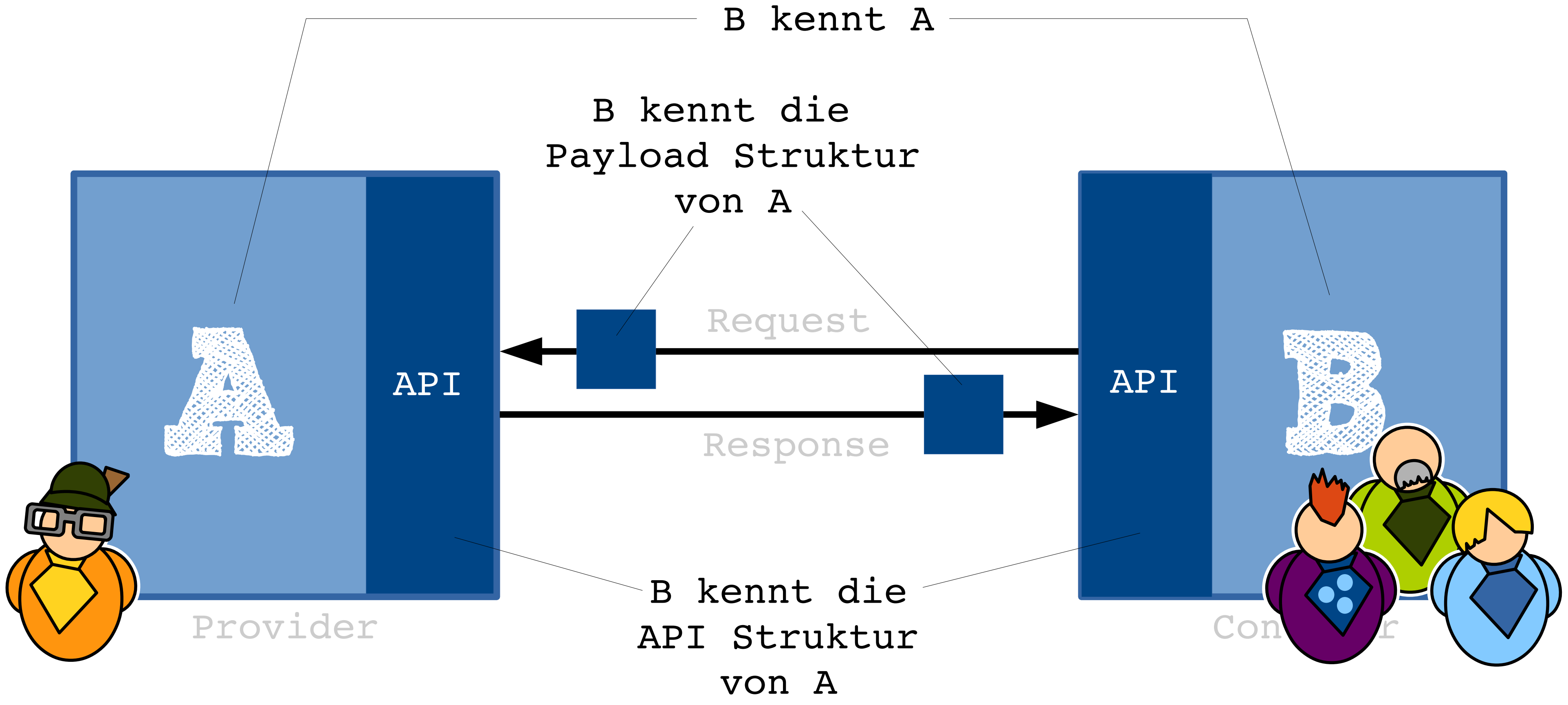


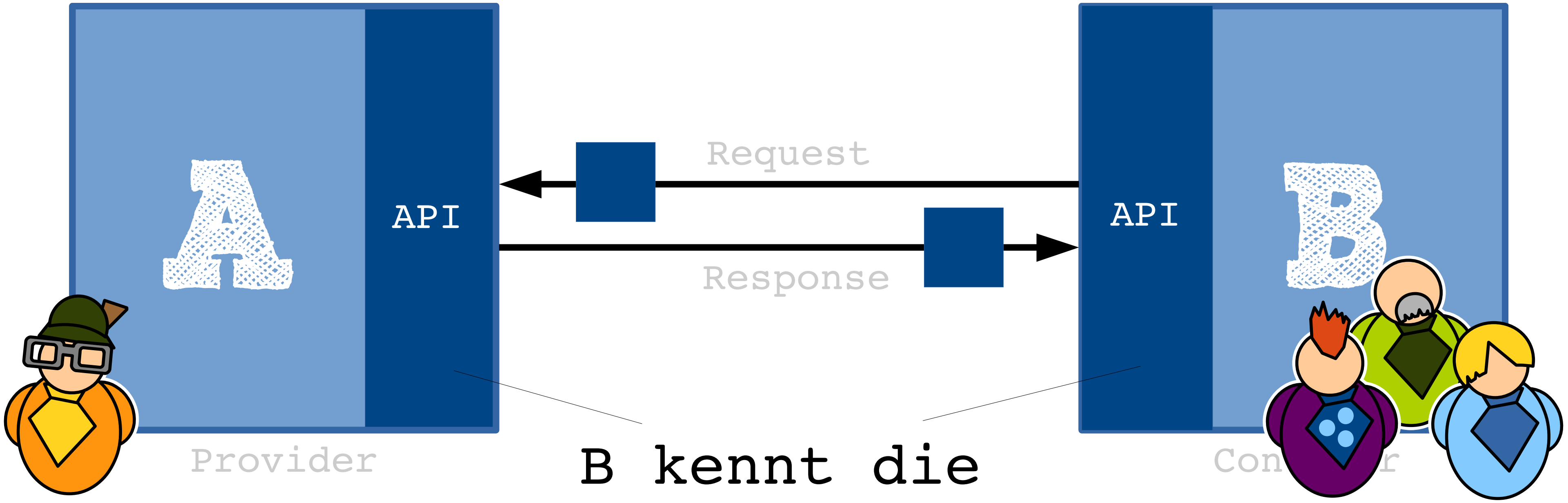
technicalDebts.xls



TECHNICAL DEPT


Payload
Request
Response
API
API
Provider
Con r


B kennt A
B kennt die Payload Struktur von A
Request
Response
A
API
API
B
Provider
Con r
B kennt die API Struktur von A



B kennt die
API Struktur
von A

RESTBUCKS, THE SAMPLE

http://codebetter.com/glennblock/2011/06/06/rest-in-practice-the-book-restbucks-the-sample

https://github.com/olivergierke/spring-restbucks

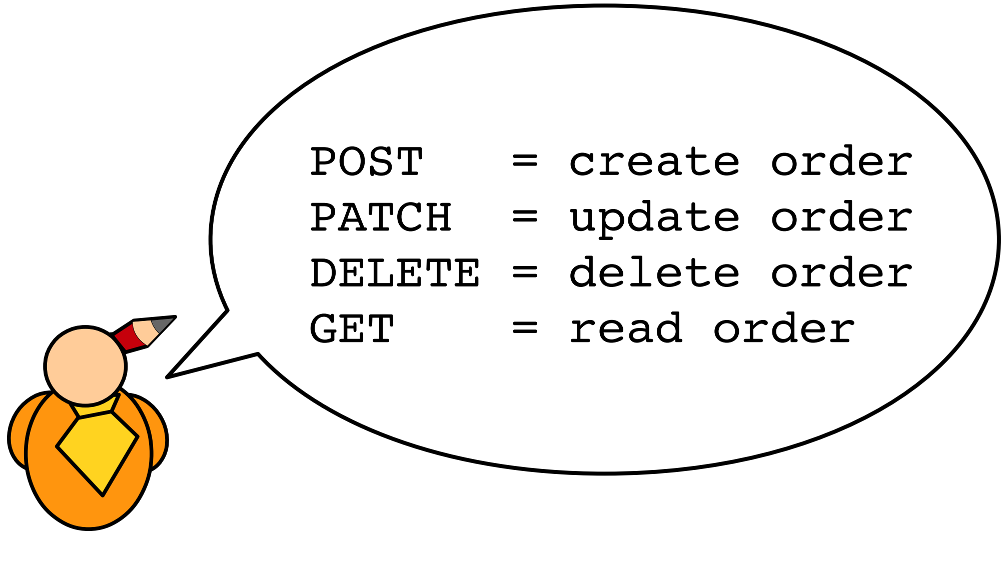
https://speakerdeck.com/olivergierke

http://olivergierke.de/2016/04/benefits-of-hypermedia/

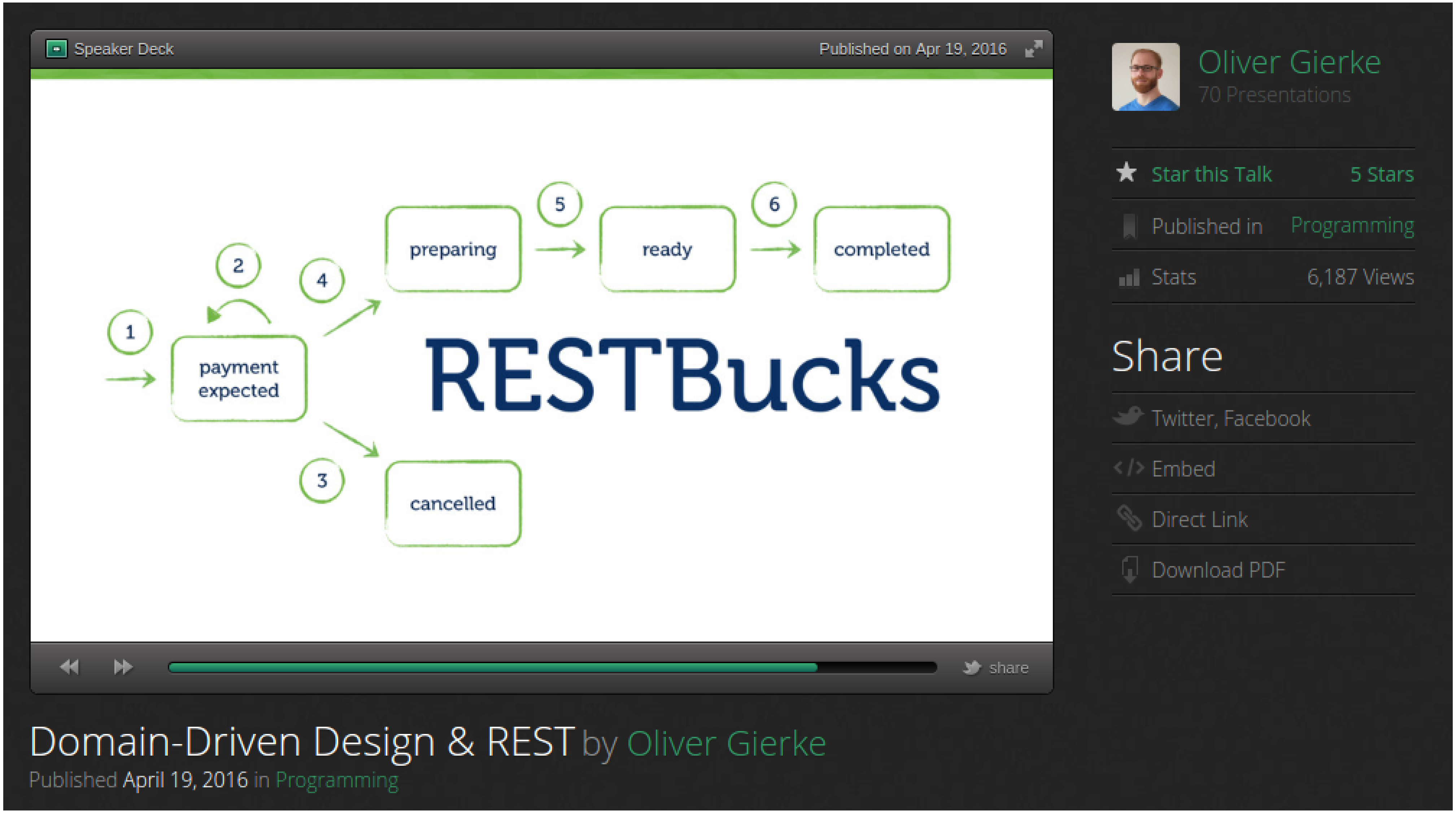
https://RESTBUCKS.IO/V1/CREATE_ORDER

https://RESTBUCKS.IO/V2/CREATE_ORDER

https://RESTBUCKS.IO/V2/ORDER/{ID}

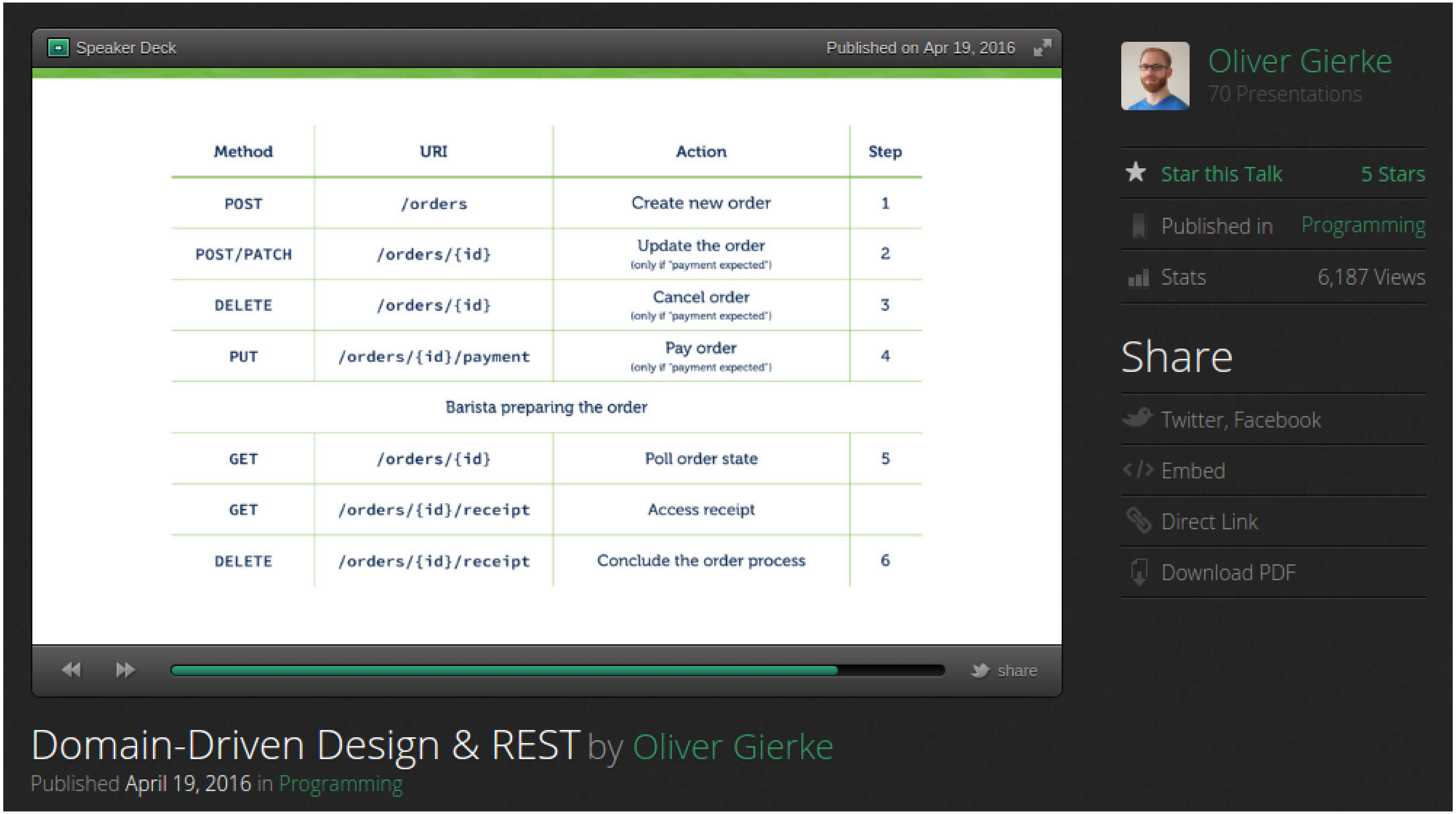


```
POST   = create order
PATCH  = update order
DELETE = delete order
GET    = read order
```



Oliver Gierke
70 Presentations

Domain-Driven Design & REST by Oliver Gierke
Published April 19, 2016 in Programming



Speaker Deck

Published on Apr 19, 2016

Method	URI	Action	Step
POST	/orders	Create new order	1
POST/PATCH	/orders/{id}	Update the order (only if "payment expected")	2
DELETE	/orders/{id}	Cancel order (only if "payment expected")	3
PUT	/orders/{id}/payment	Pay order (only if "payment expected")	4
		Barista preparing the order	
GET	/orders/{id}	Poll order state	5
GET	/orders/{id}/receipt	Access receipt	
DELETE	/orders/{id}/receipt	Conclude the order process	6

Domain-Driven Design & REST by Oliver Gierke

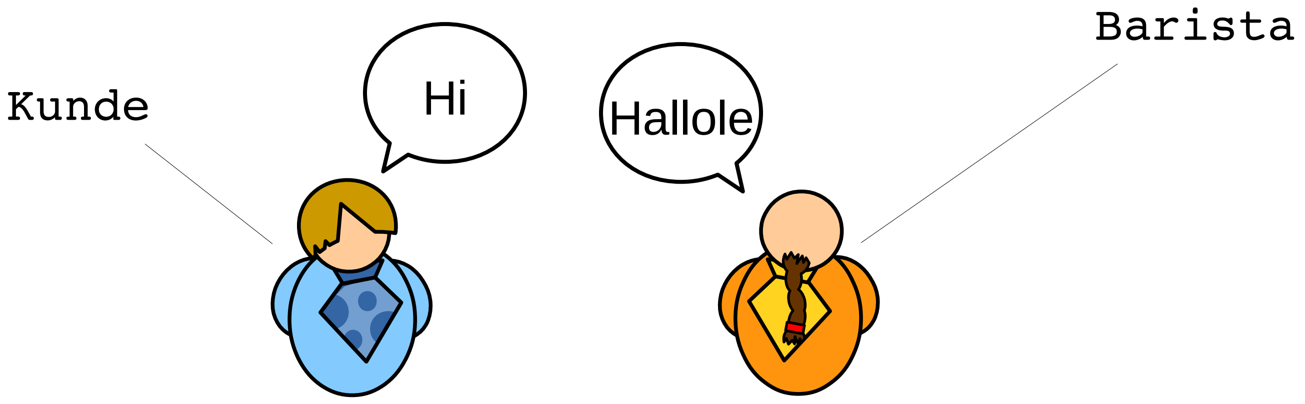
Published April 19, 2016 in Programming

Oliver Gierke

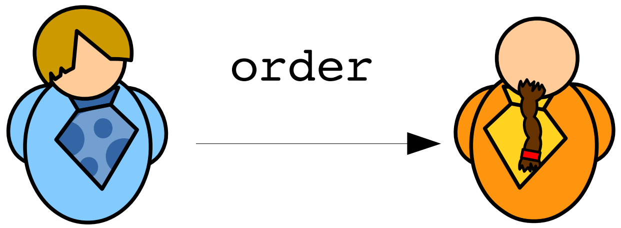
70 Presentations

Star this Talk	5 Stars
Published in	Programming
Stats	6,187 Views

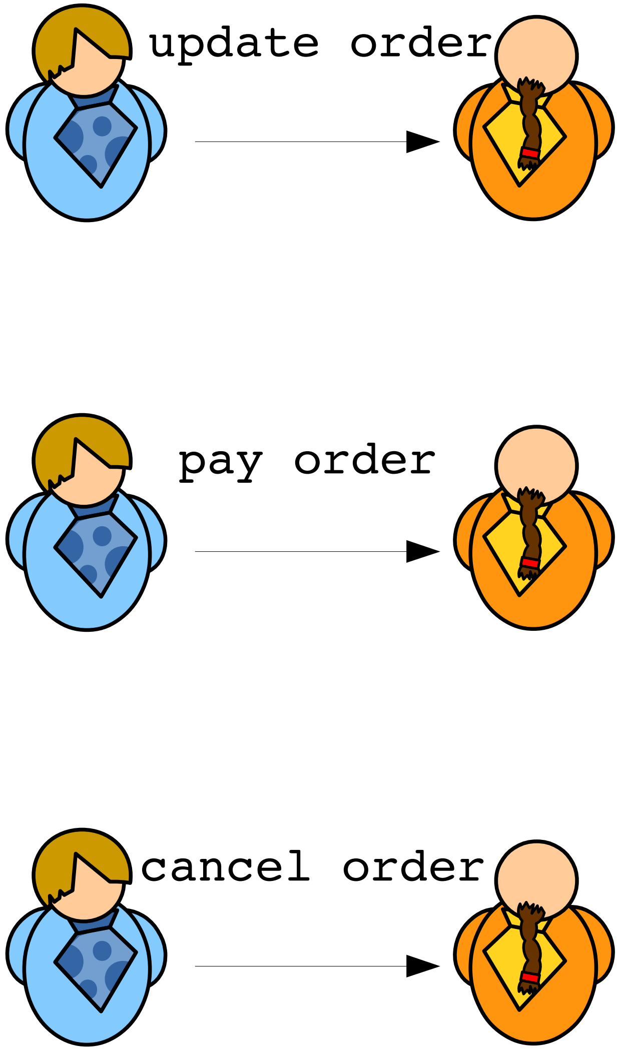
Share


Kunde
Hi
Hallole
Barista

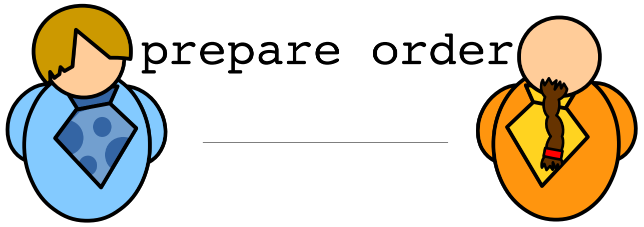
SCHRITT 1



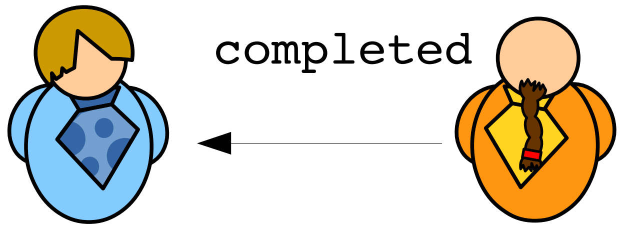
SCHRITT 2



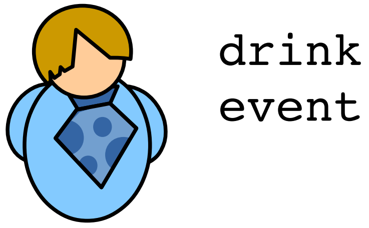
SCHRITT 3



SCHRITT 4



SCHRITT 5

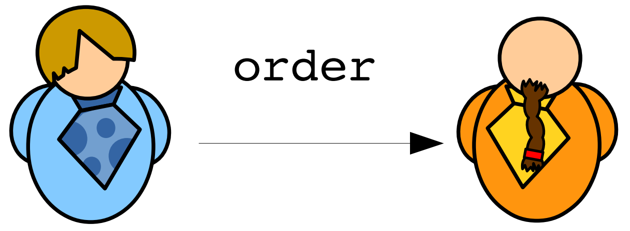


HATEOAS

Hypermedia
As
The
Engine
Of
Application
State

SCHRITT 1

Request	Response

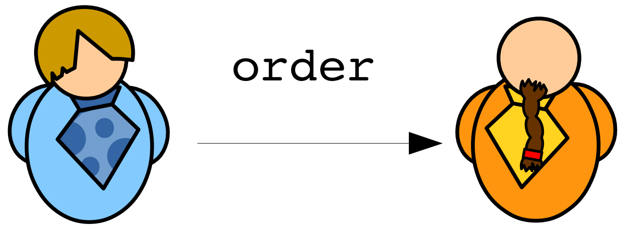


POST: RESTBUCKS.io/v2/order

```
self
GET:         RESTBUCKS.io/v2/order/1
             RESTBUCKS.io/v2/order/{id}

update_order
POST/PATCH:  RESTBUCKS.io/v2/order/1
             RESTBUCKS.io/v2/order/{id}

cancel_order
DELETE:      RESTBUCKS.io/v2/order/1
             RESTBUCKS.io/v2/order/{id}

pay_order
PUT:         RESTBUCKS.io/v2/order/1/payment
             RESTBUCKS.io/v2/order/{id}/payment
```

SCHRITT 1

Request	Response



POST: RESTBUCKS.io/v2/order

self

```
GET:          RESTBUCKS.io/v2/order/1
              RESTBUCKS.io/v2/order/{id}
```

update_order

```
POST/PATCH:   RESTBUCKS.io/v2/order/1
              RESTBUCKS.io/v2/order/{id}
```

cancel_order

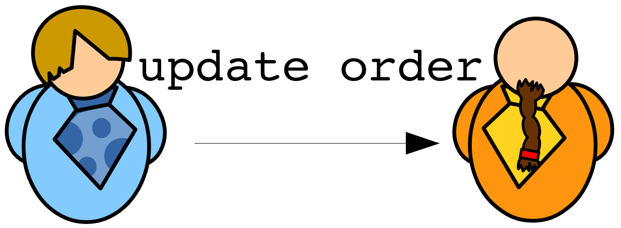
```
DELETE:       RESTBUCKS.io/v2/order/1
              RESTBUCKS.io/v2/order/{id}
```

pay_order

```
PUT:          RESTBUCKS.io/v2/order/1/payment
              RESTBUCKS.io/v2/order/{id}/payment
```

SCHRITT 2

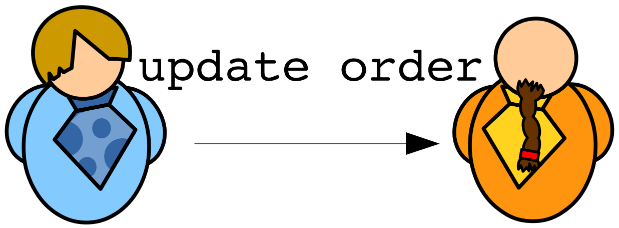
Request	Response



PATCH: RESTBUCKS.io/v2/order/1
RESTBUCKS.io/v2/order/{id}

SCHRITT 2

Request	Response



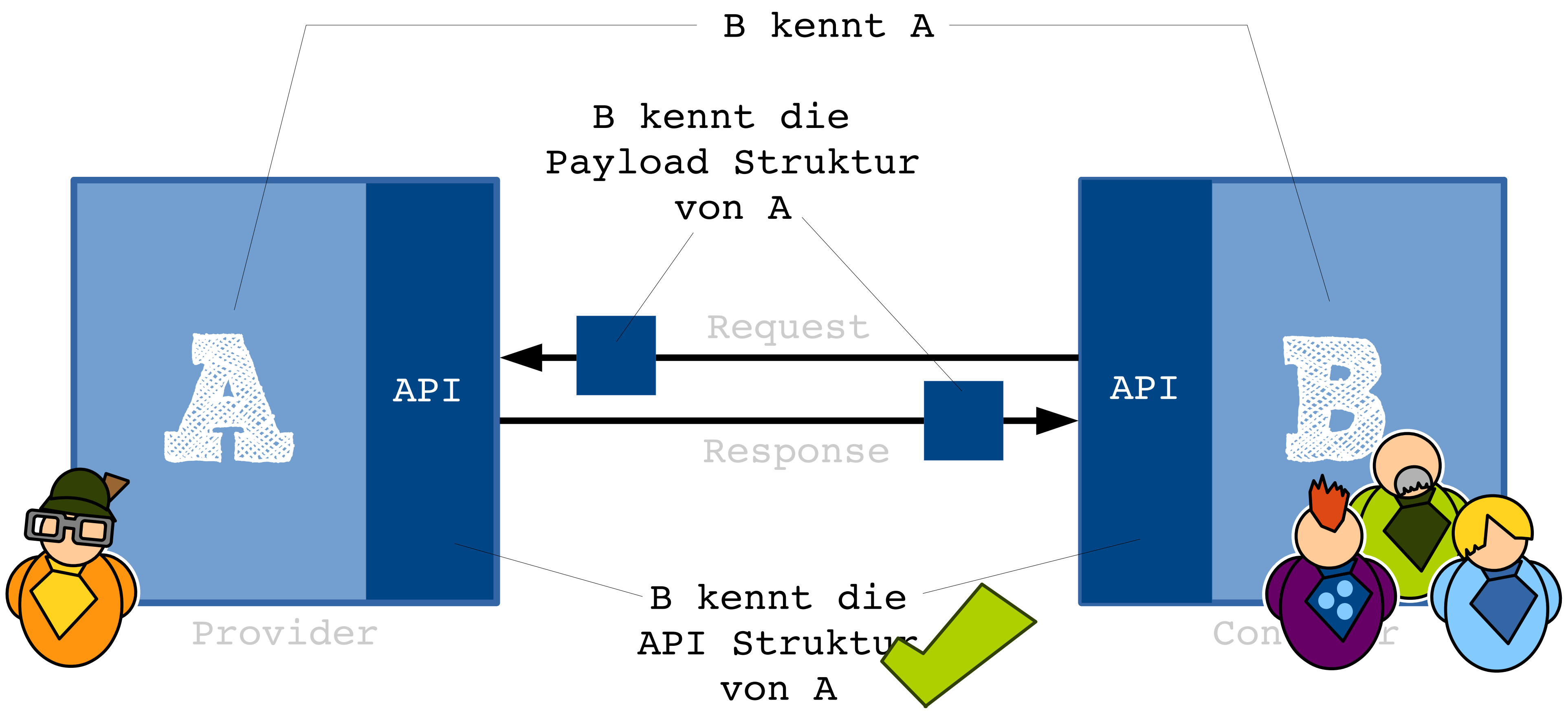
PATCH: RESTBUCKS.io/v2/order/1
       RESTBUCKS.io/v2/order/{id}

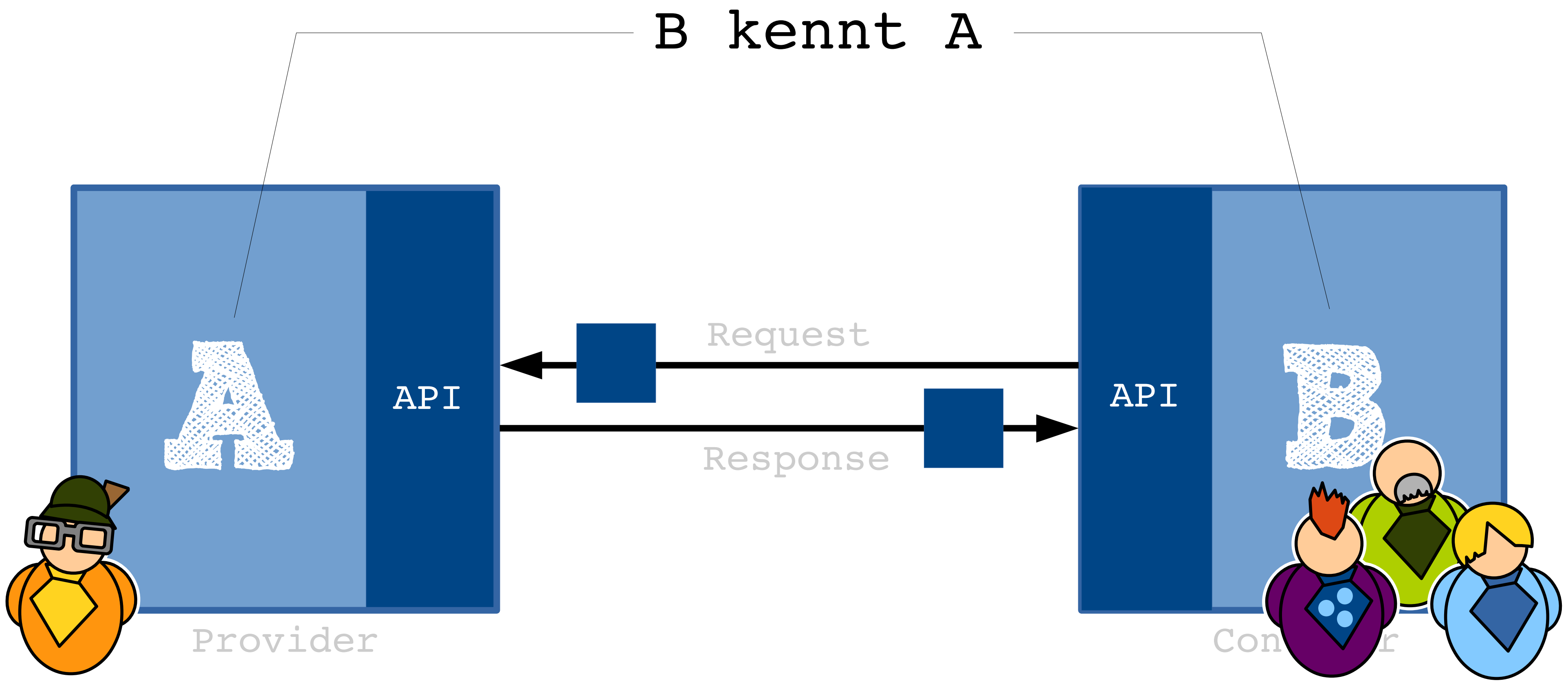
self
```
GET:         RESTBUCKS.io/v2/order/1
             RESTBUCKS.io/v2/order/{id}
```

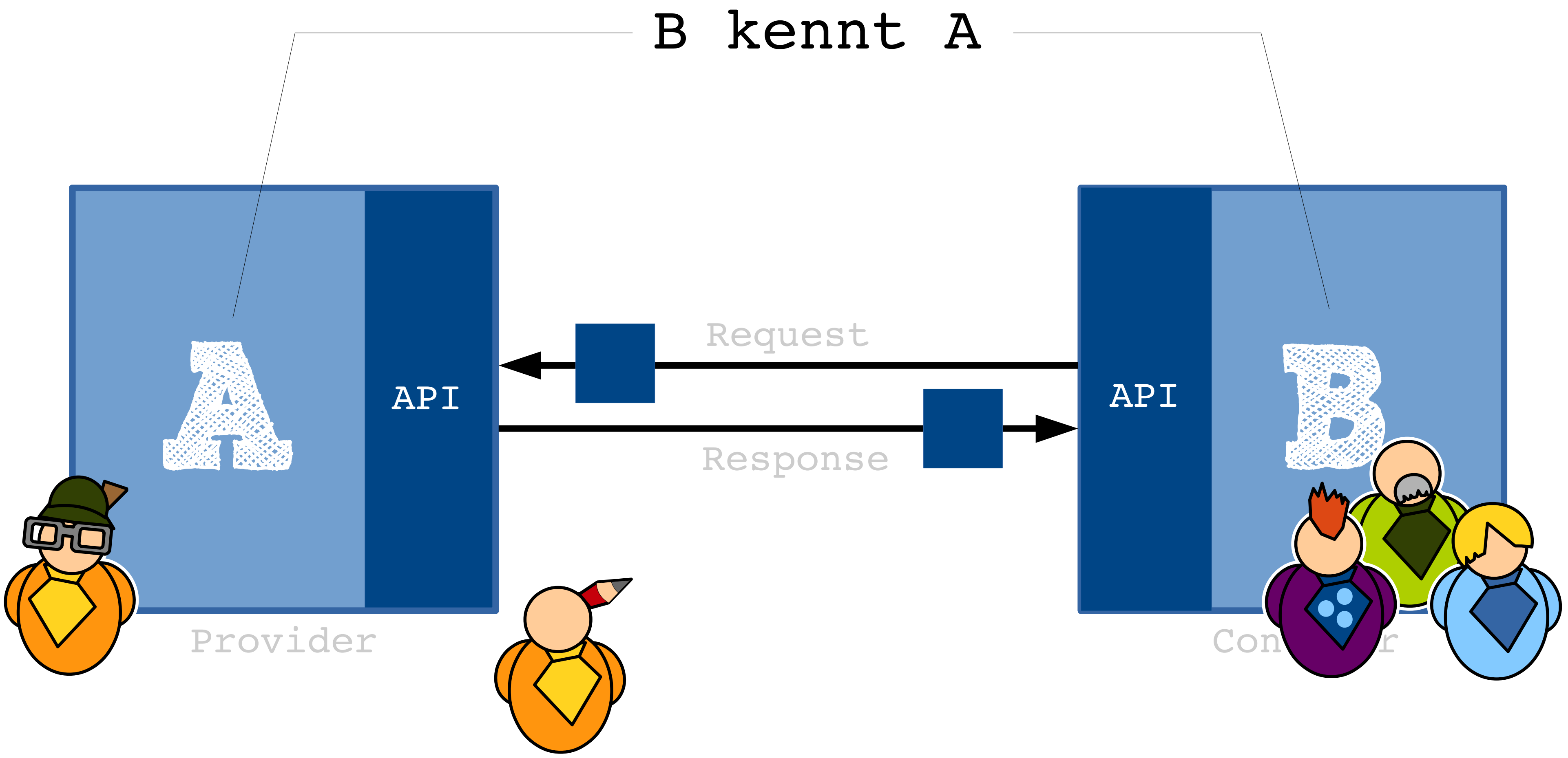
update_order
```
POST/PATCH:  RESTBUCKS.io/v2/order/1
             RESTBUCKS.io/v2/order/{id}
```

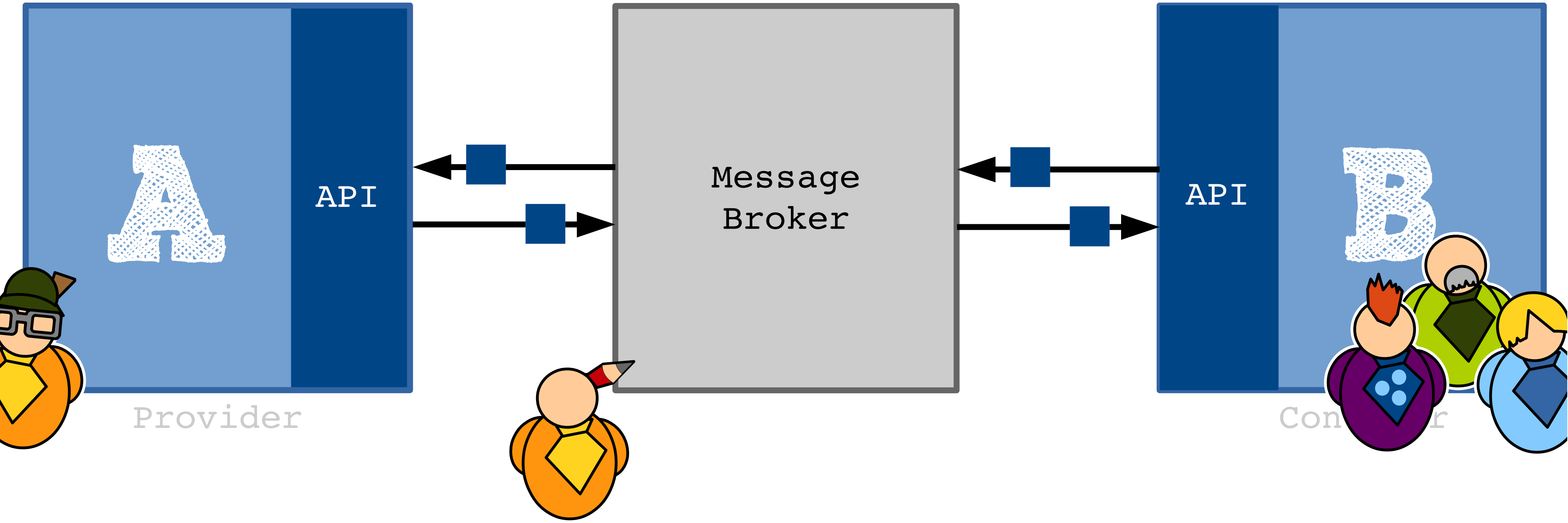
cancel_order
```
DELETE:      RESTBUCKS.io/v2/order/1
             RESTBUCKS.io/v2/order/{id}
```

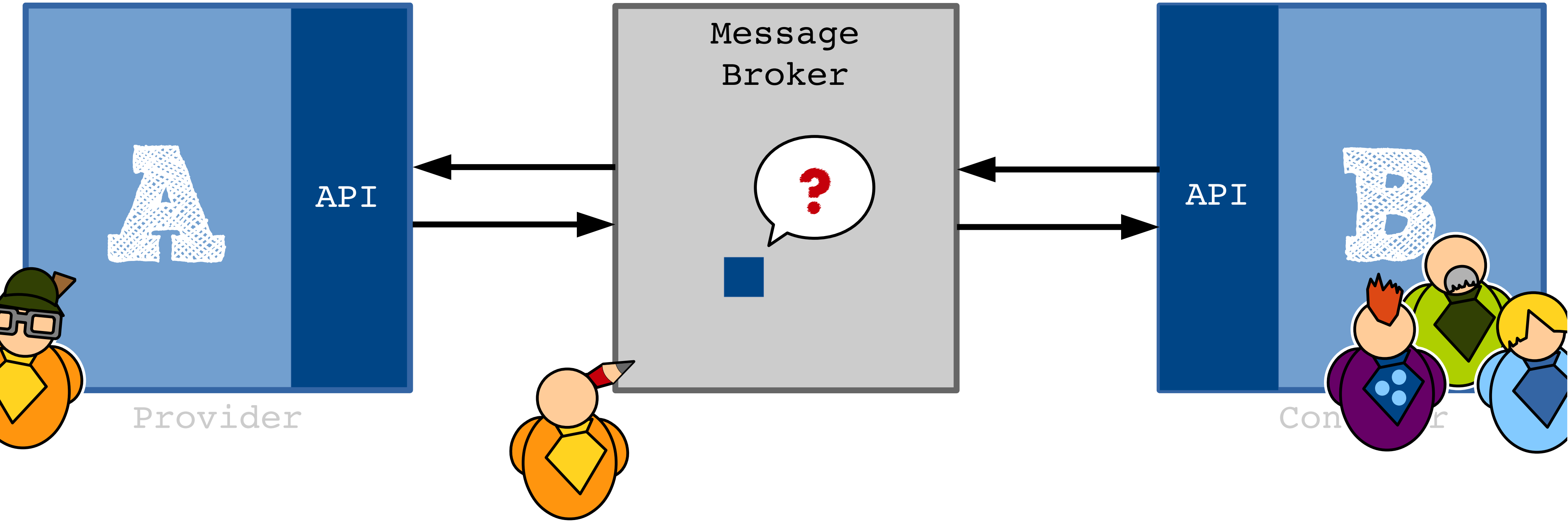
pay_order
```
PUT:         RESTBUCKS.io/v2/order/1/payment
             RESTBUCKS.io/v2/order/{id}/payment
```


B kennt A
B kennt die Payload Struktur von A
Request
A
API
Response
API
B
Provider
Consumer
B kennt die API Struktur von A

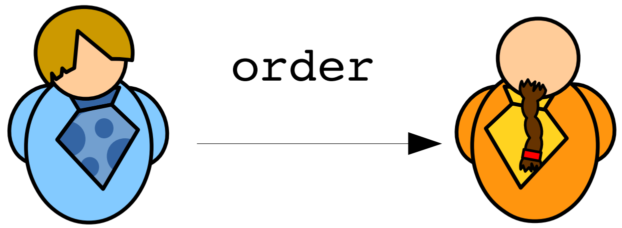

B kennt A
A
API
Request
Response
API
B
Provider
Con r


B kennt A
A
API
Request
Response
API
B
Provider
Con r

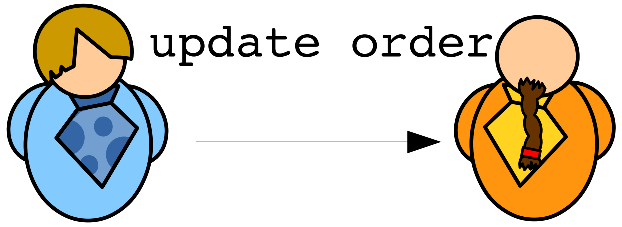

A
API
Provider
Message Broker
API
B
Con r

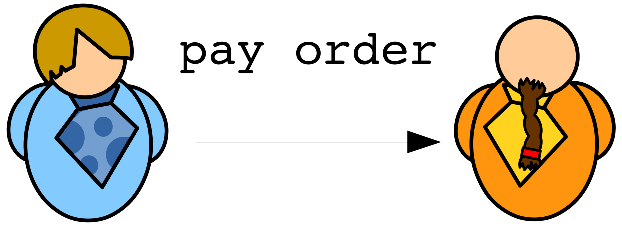

A
API
Provider
Message
Broker
?
API
B
Con r

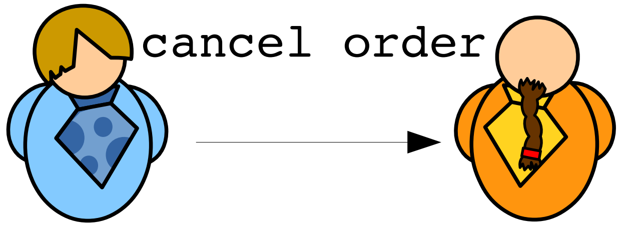
SCHRITT 1



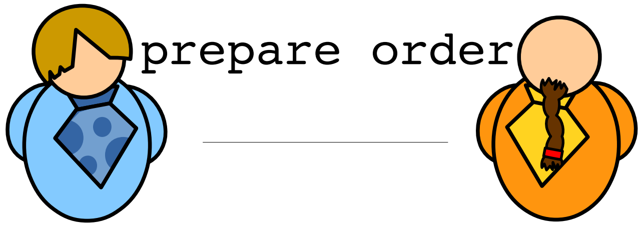
SCHRITT 2



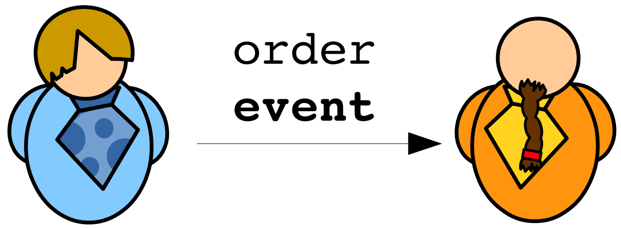




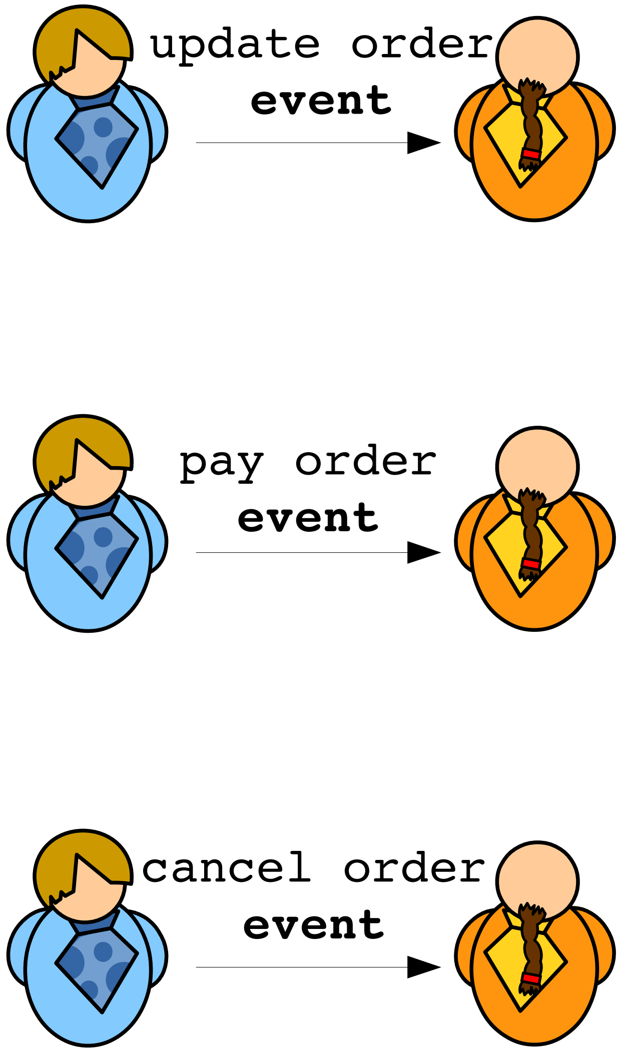
SCHRITT 3



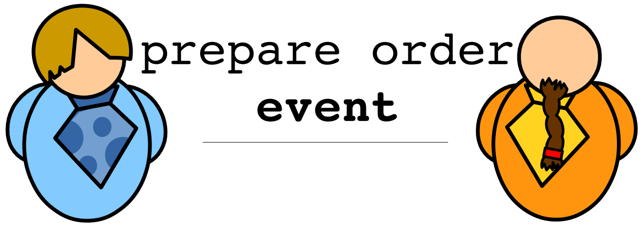
SCHRITT 1

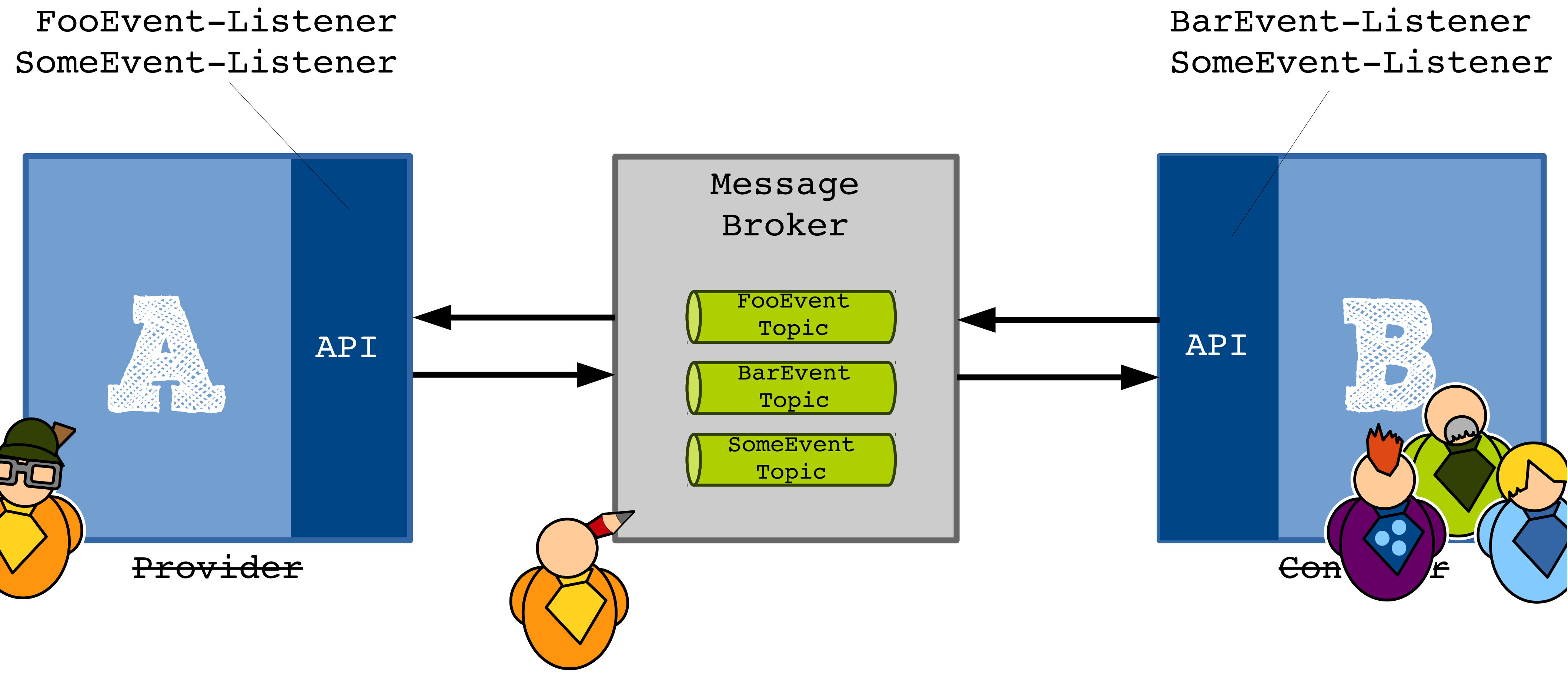


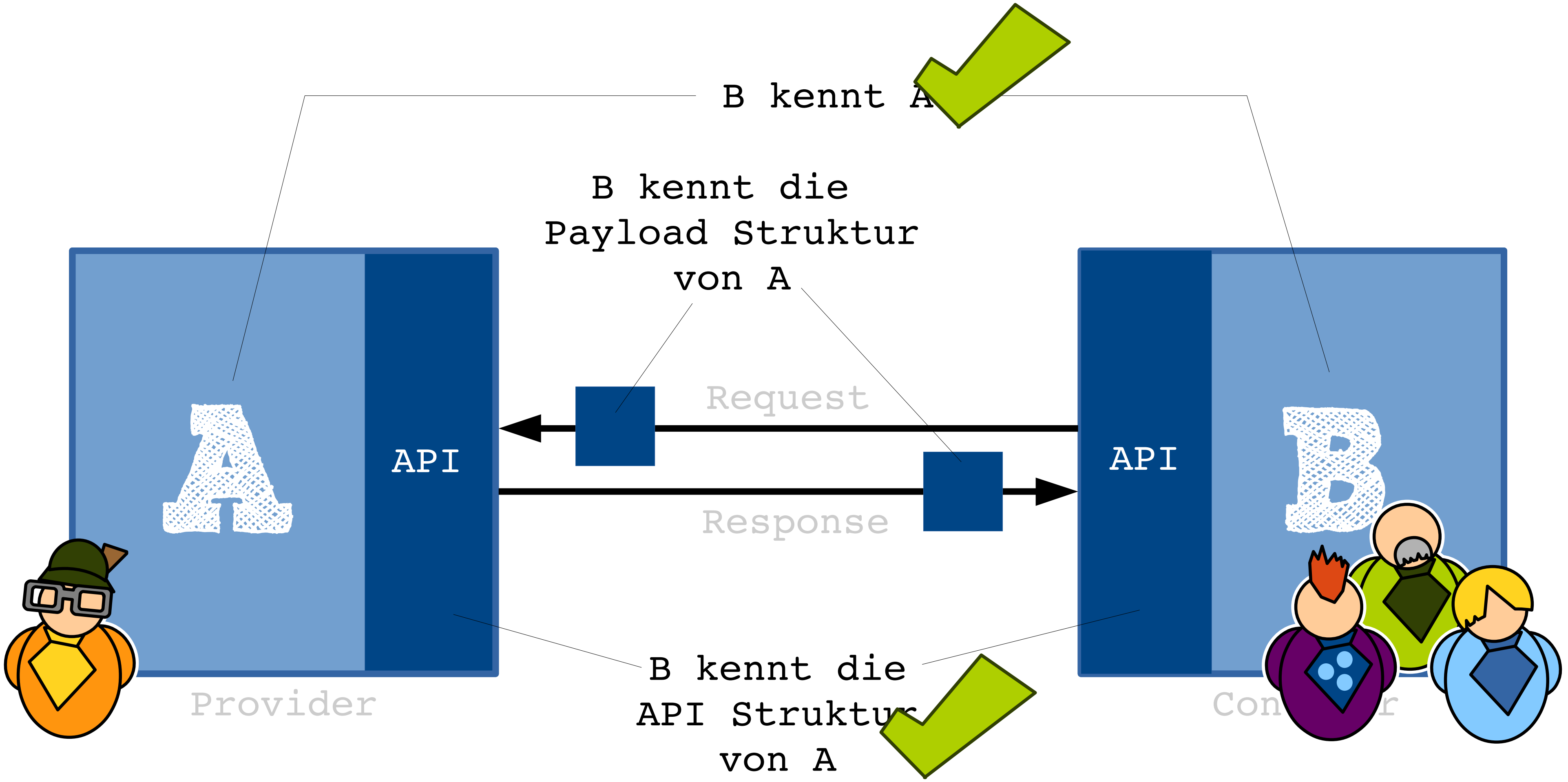
SCHRITT 2



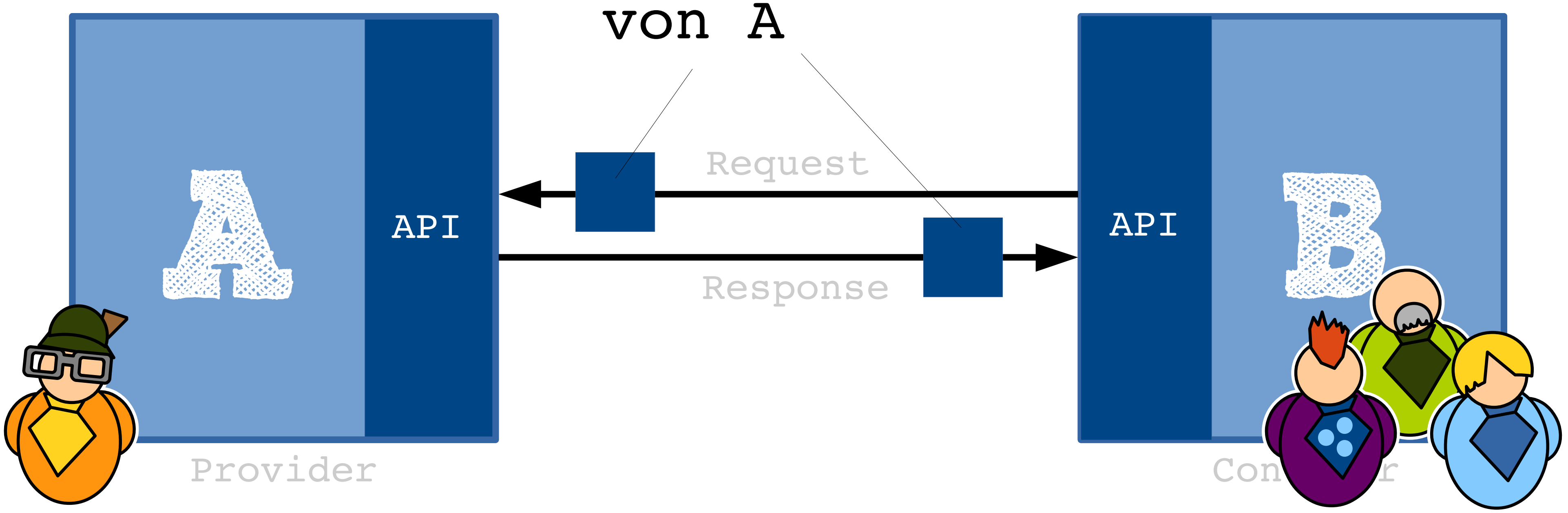
SCHRITT 3



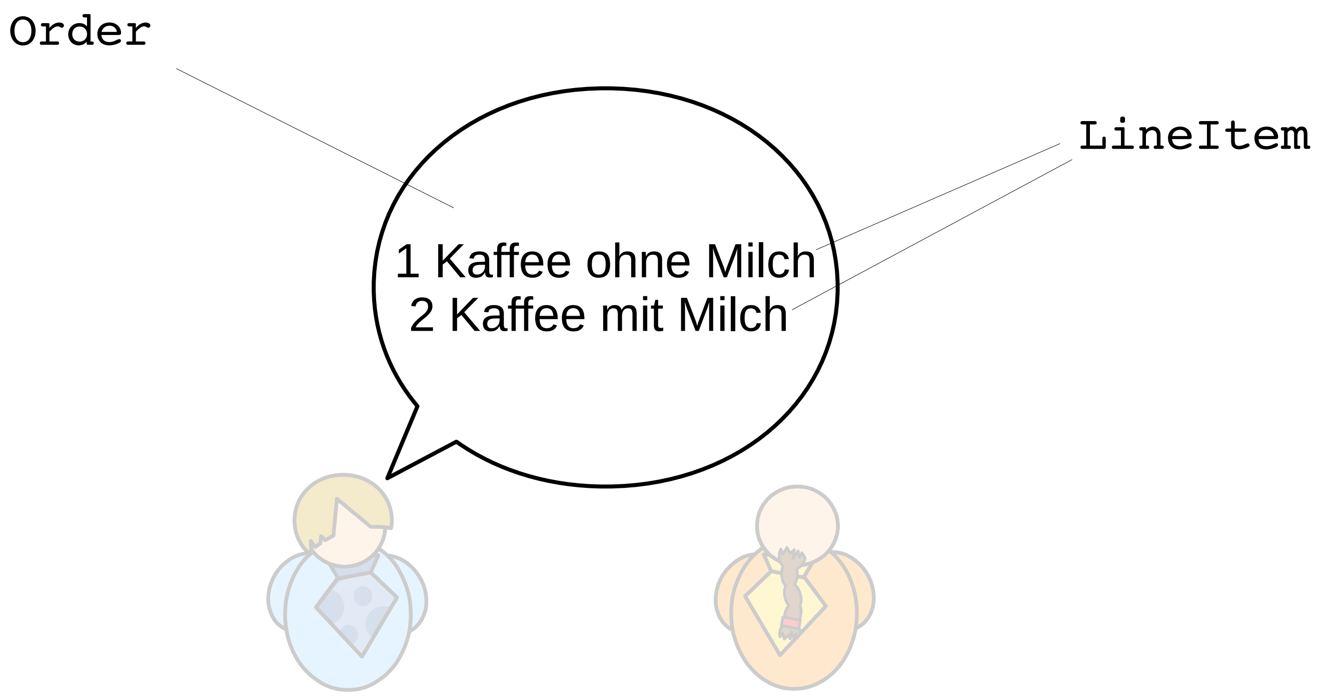

FooEvent-Listener
SomeEvent-Listener
BarEvent-Listener
SomeEvent-Listener
A
API
Message Broker
FooEvent Topic
BarEvent Topic
SomeEvent Topic
API
B
~~Provider~~
~~Con r~~

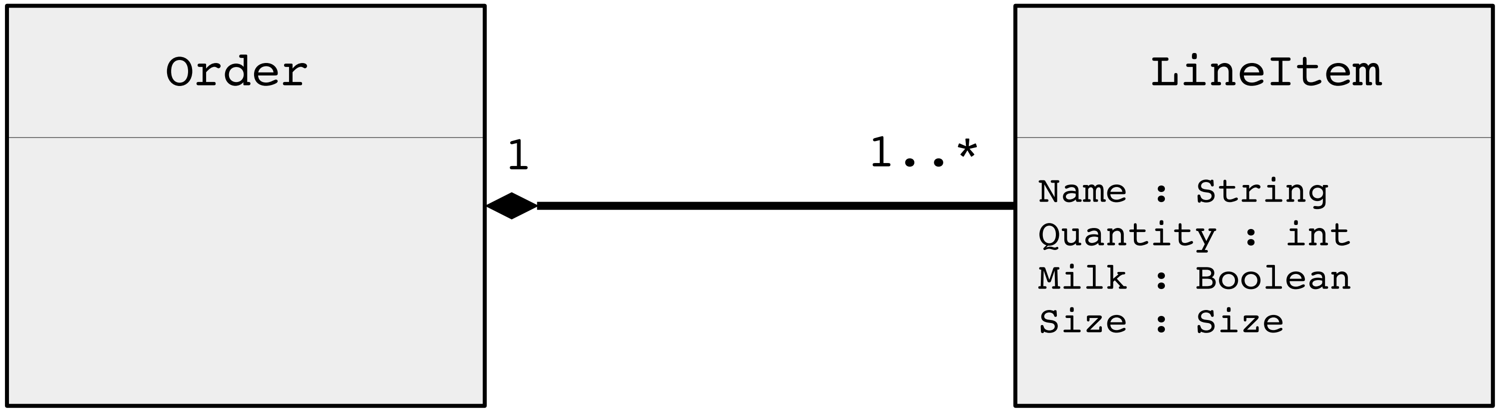

B kennt A
B kennt die Payload Struktur von A
Request
Response
A
API
API
B
Provider
B kennt die API Struktur von A

B kennt die Payload Struktur von A



1 Kaffee ohne Milch
2 Kaffee mit Milch

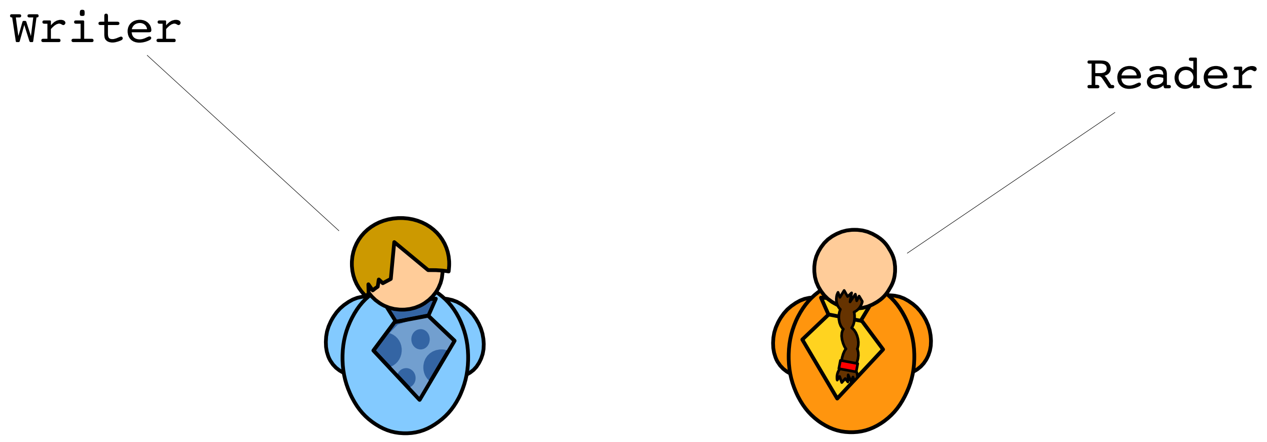

Order
LineItem
1 Kaffee ohne Milch
2 Kaffee mit Milch

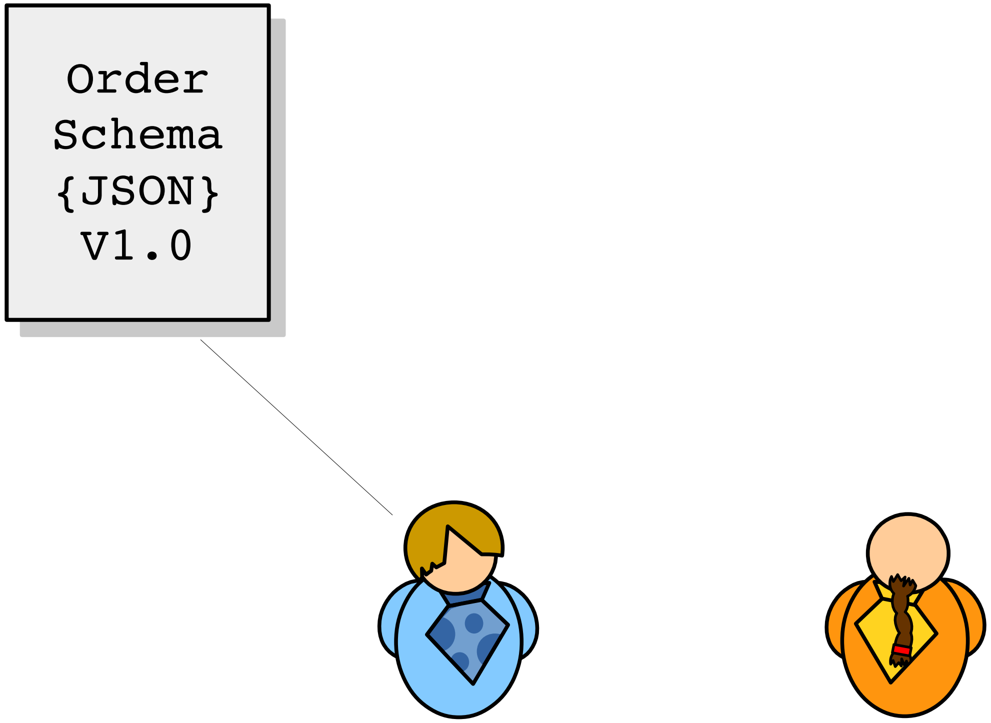

Order
1
1..*
LineItem
Name : String
Quantity : int
Milk : Boolean
Size : Size

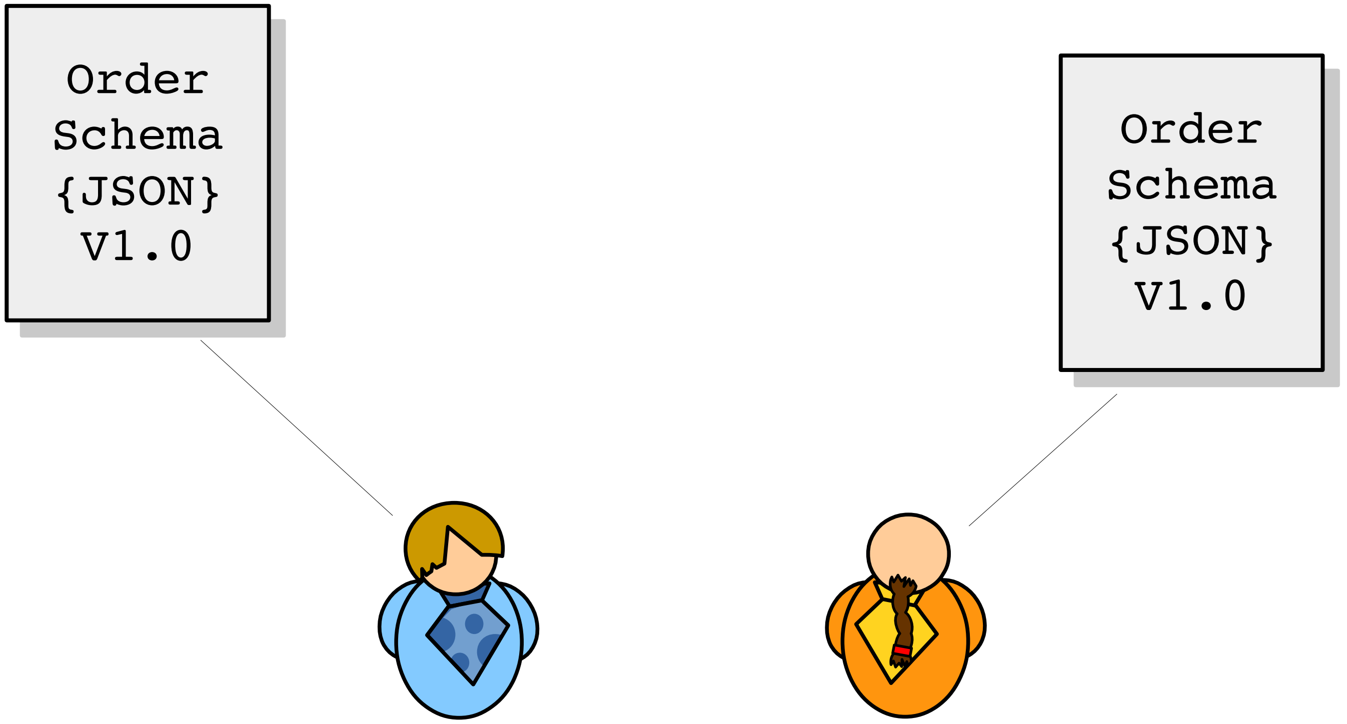
SCHEMA EVOLUTION

MY HADOOP IS BIGGER THAN YOURS...

AVRO™


Writer
Reader


Order
Schema
{JSON}
V1.0


Order Schema {JSON} V1.0
Order Schema {JSON} V1.0

Version 1.0

```
[
    {
        "type": "enum",
        "name": "Sizes",
        "symbols": ["LARGE", "MIDDLE", "SMALL"]
    },
    {
        "type": "record",
        "name": "LineItem",
        "fields": [
            {"name": "name", "type": "string"},
            {"name": "quantity", "type": "int"},
            {"name": "milk", "type": "boolean"},
            {"name": "size", "type": "Sizes"}
        ]
    },
    {
        "type": "record",
        "name": "Order",
        "fields": [
            {"name": "item", "type": "LineItem"}
        ]
    }

]
```

Version 1.0

```
[
    {
        "type": "enum",
        "name": "Sizes",
        "symbols": ["LARGE", "MIDDLE", "SMALL"]
    },
    {
        "type": "record",
        "name": "LineItem",
        "fields": [
            {"name": "name", "type": "string"},
            {"name": "quantity", "type": "int"},
            {"name": "milk", "type": "boolean"},
            {"name": "size", "type": "Sizes"}
        ]
    },
    {
        "type": "record",
        "name": "Order",
        "fields": [
            {"name": "item", "type": "LineItem"}
        ]
    }

]
```

Enumeration

Version 1.0

```
[
    {
        "type": "enum",
        "name": "Sizes",
        "symbols": ["LARGE", "MIDDLE", "SMALL"]
    },
    {
        "type": "record",
        "name": "LineItem",
        "fields": [
            {"name": "name", "type": "string"},
            {"name": "quantity", "type": "int"},
            {"name": "milk", "type": "boolean"},
            {"name": "size", "type": "Sizes"}
        ]
    },
    {
        "type": "record",
        "name": "Order",
        "fields": [
            {"name": "item", "type": "LineItem"}
        ]
    }

]
```

LineItem Type

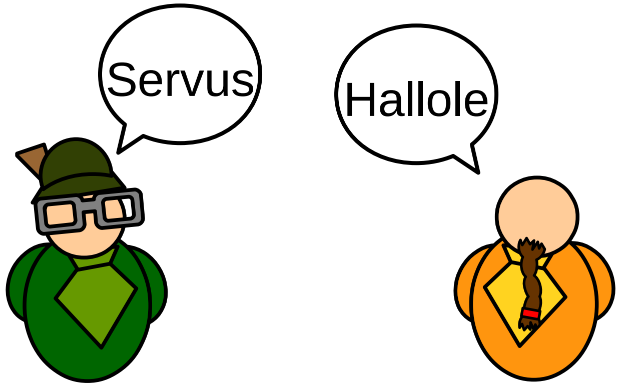
Version 1.0

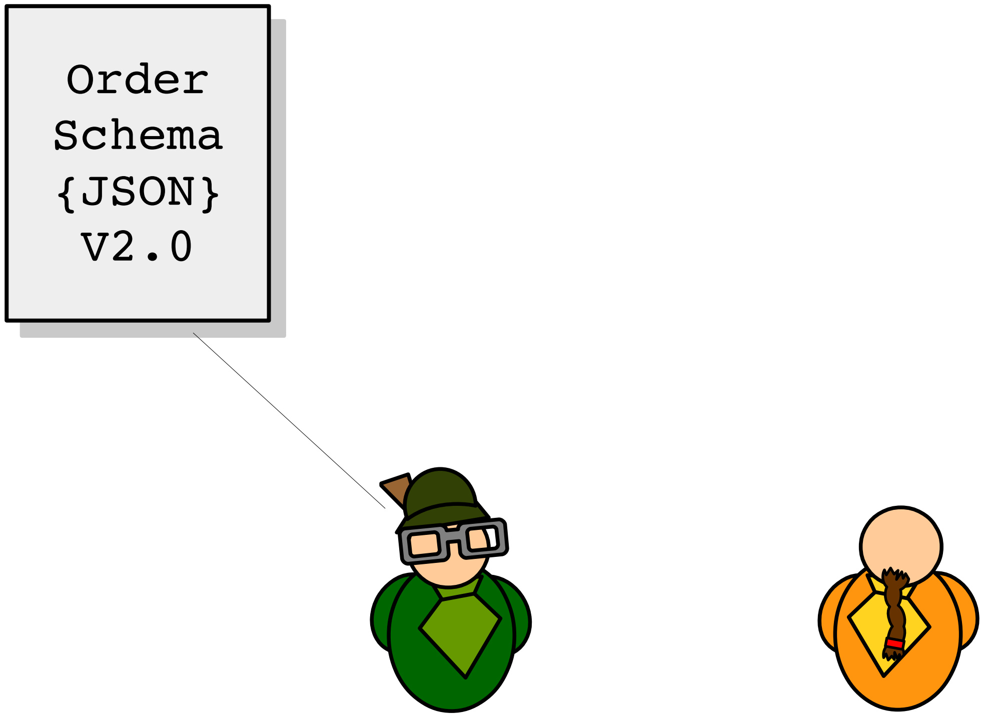
```
[
    {
        "type": "enum",
        "name": "Sizes",
        "symbols": ["LARGE", "MIDDLE", "SMALL"]
    },
    {
        "type": "record",
        "name": "LineItem",
        "fields": [
            {"name": "name", "type": "string"},
            {"name": "quantity", "type": "int"},
            {"name": "milk", "type": "boolean"},
            {"name": "size", "type": "Sizes"}
        ]
    },
    {
        "type": "record",
        "name": "Order",
        "fields": [
            {"name": "item", "type": "LineItem"}
        ]
    }

]
```

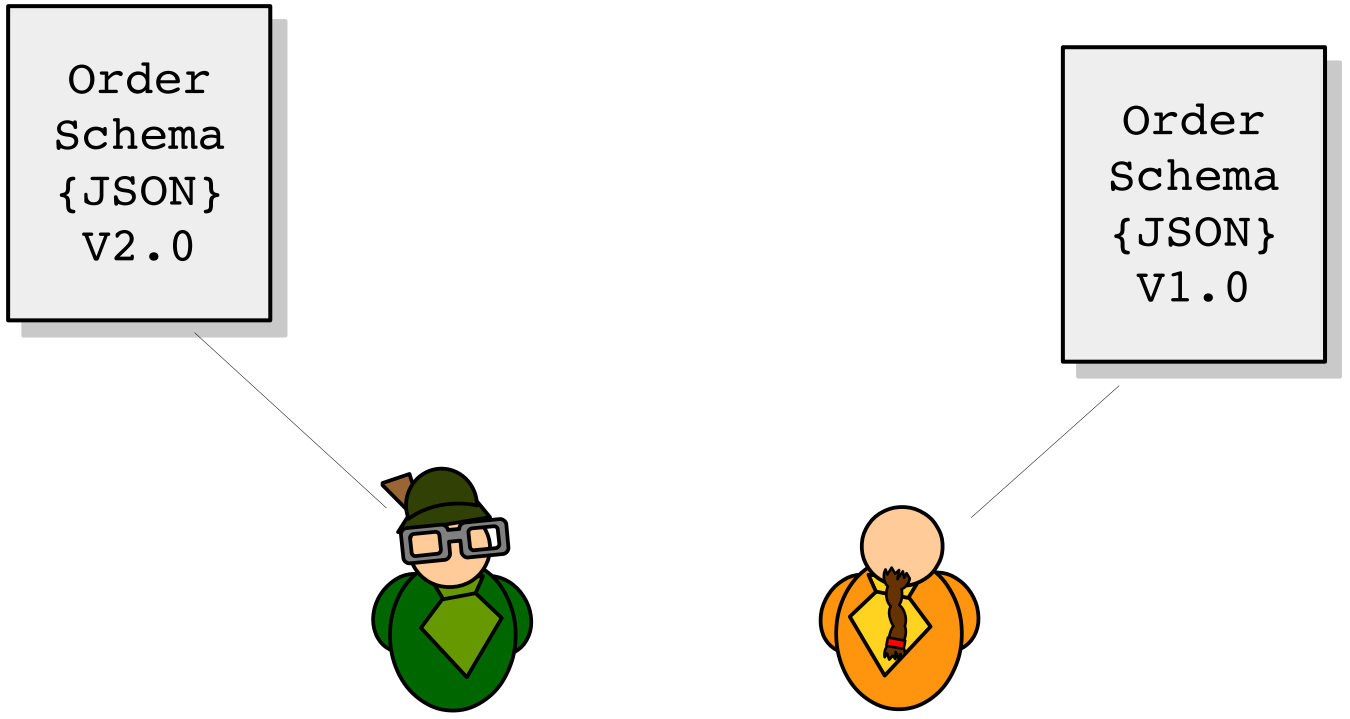
Order Type

Version 1.0

```
[
    {
        "type": "enum",
        "name": "Sizes",
        "symbols": ["LARGE", "MIDDLE", "SMALL"]
    },
    {
        "type": "record",
        "name": "LineItem",
        "fields": [
            {"name": "name", "type": "string"},
            {"name": "quantity", "type": "int"},
            {"name": "milk", "type": "boolean"},
            {"name": "size", "type": "Sizes"}
        ]
    },
    {
        "type": "record",
        "name": "Order",
        "fields": [
            {"name": "item", "type": "LineItem"}
        ]
    }

]
```

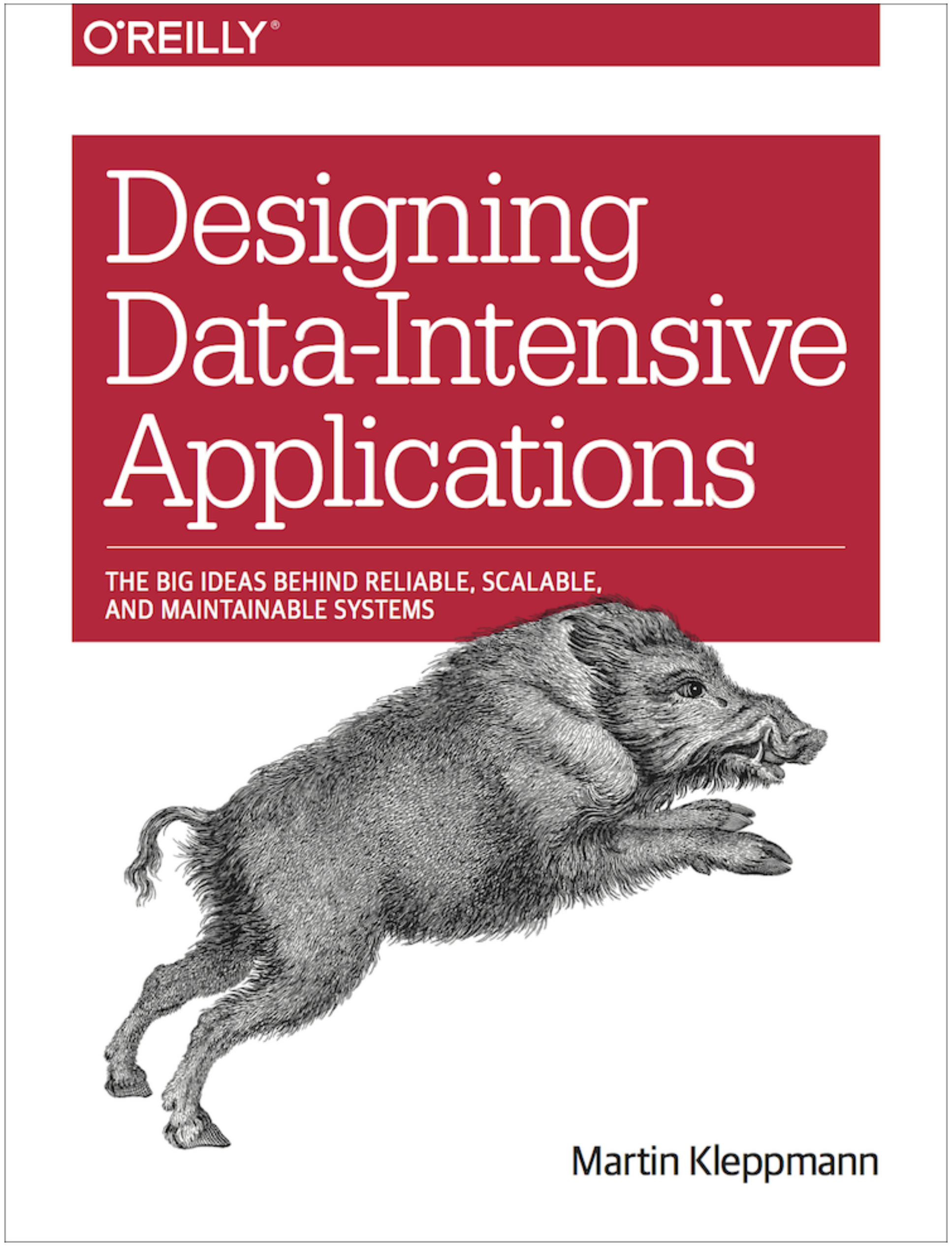

Servus
Hallole


Order
Schema
{JSON}
V2.0


Order Schema {JSON} V2.0
Order Schema {JSON} V1.0

Version 2.0

```
[
    {
        "type": "enum",
        "name": "Sizes",
        "symbols": ["LARGE", "MIDDLE", "SMALL"]
    },
    {
        "type": "record",
        "name": "LineItem",
        "fields": [
            {"name": "name", "type": "string"},
            {"name": "quantity", "type": "int"},
            {"name": "milk", "type": "boolean"},
            {"name": "size", "type": "Sizes"}
        ]
    },
    {
        "type": "record",
        "name": "Order",
        "fields": [
            {"name": "item", "type": "LineItem"}
        ]
    }

]
```



O'REILLY®

Designing Data-Intensive Applications

THE BIG IDEAS BEHIND RELIABLE, SCALABLE, AND MAINTAINABLE SYSTEMS

Martin Kleppmann

Schema Registry Server:

spring-cloud-stream-schema-server als Dependency

@EnableSchemaRegistryServer

Schema Registry Client:

spring-cloud-stream-schema als Dependency

@EnableSchemaRegistryClient

Funktioniert auch mit der confluent schema registry (https://www.confluent.io/)