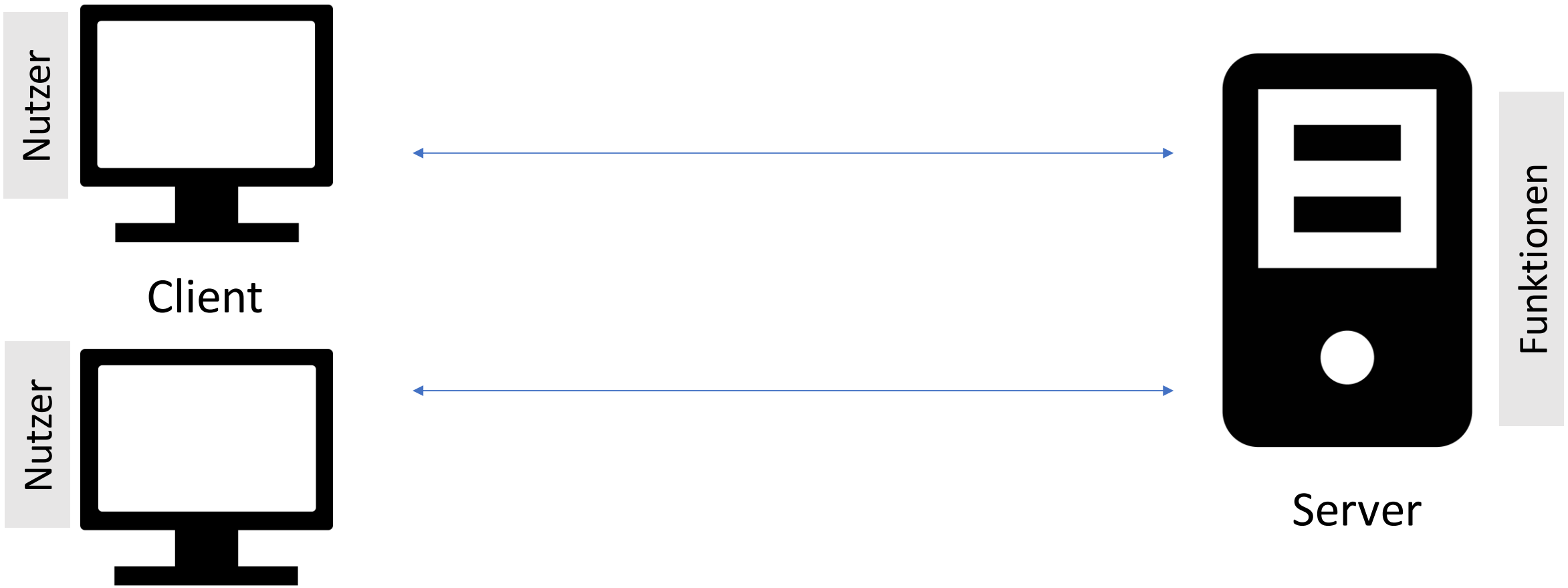
RESTful Processes

Extending the OpenAPI Specification to Enable the Documentation of RESTful Processes

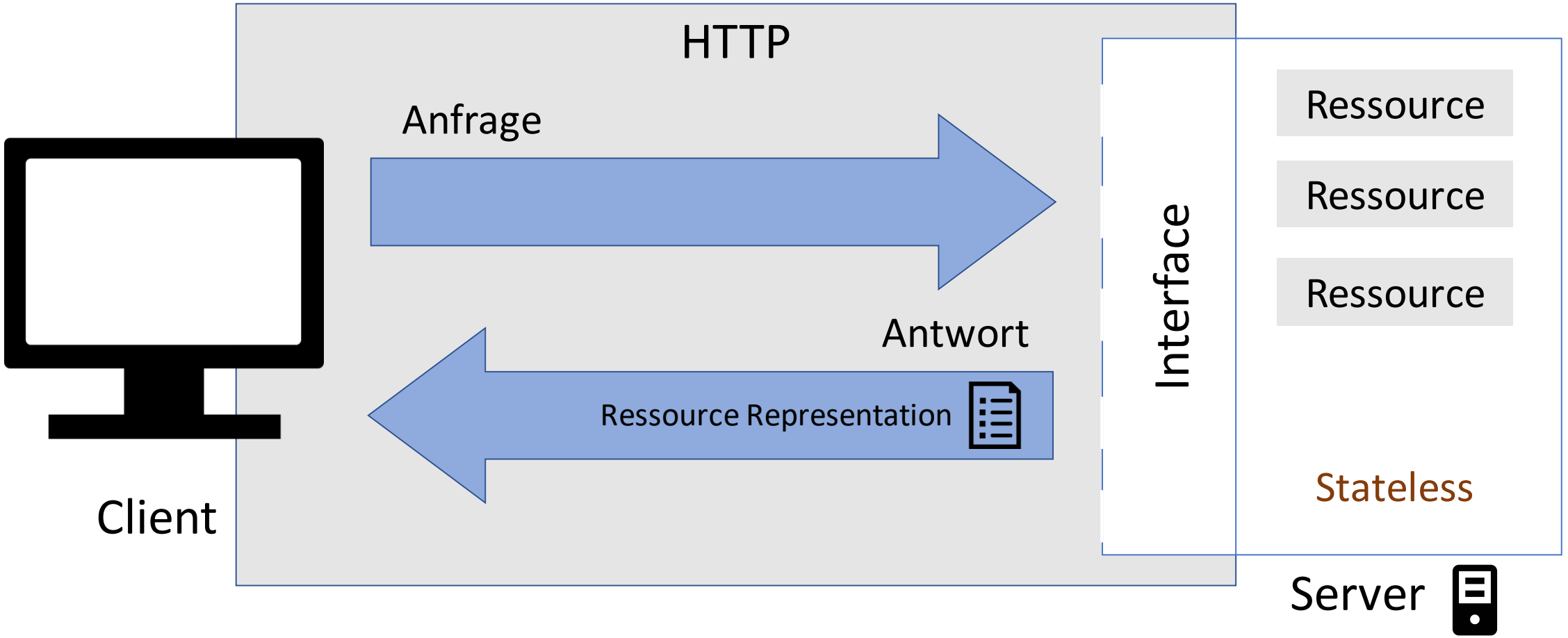
28.09.2018
DevCamp 2018, Berlin

AccessHolding MICROFINANCE AG Steffen Viebrock

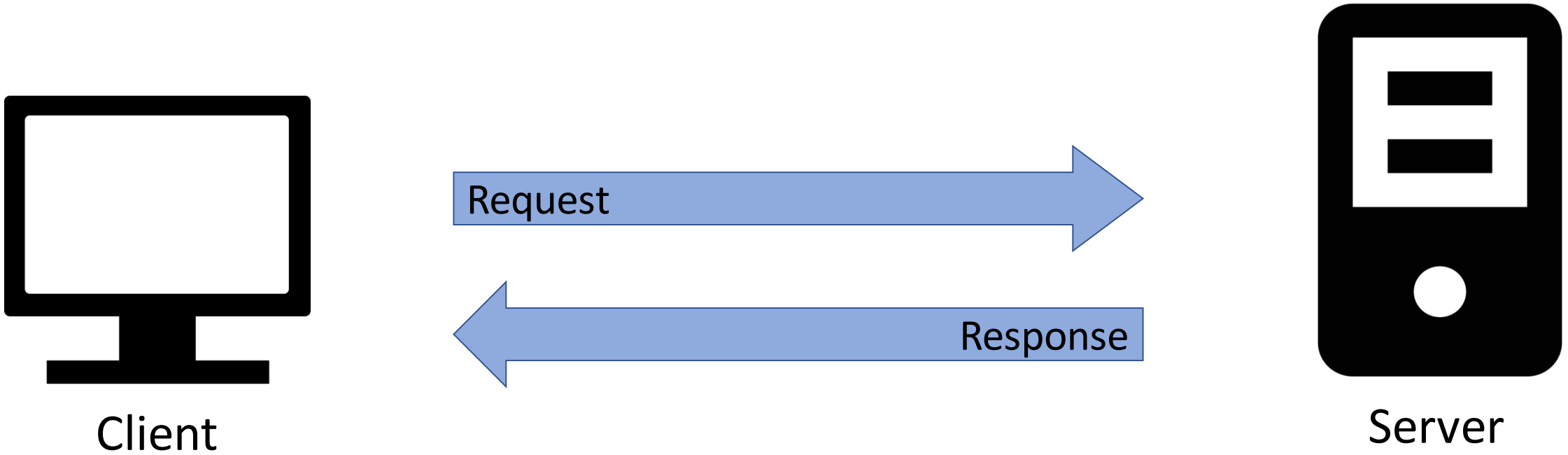
Client Server Architektur



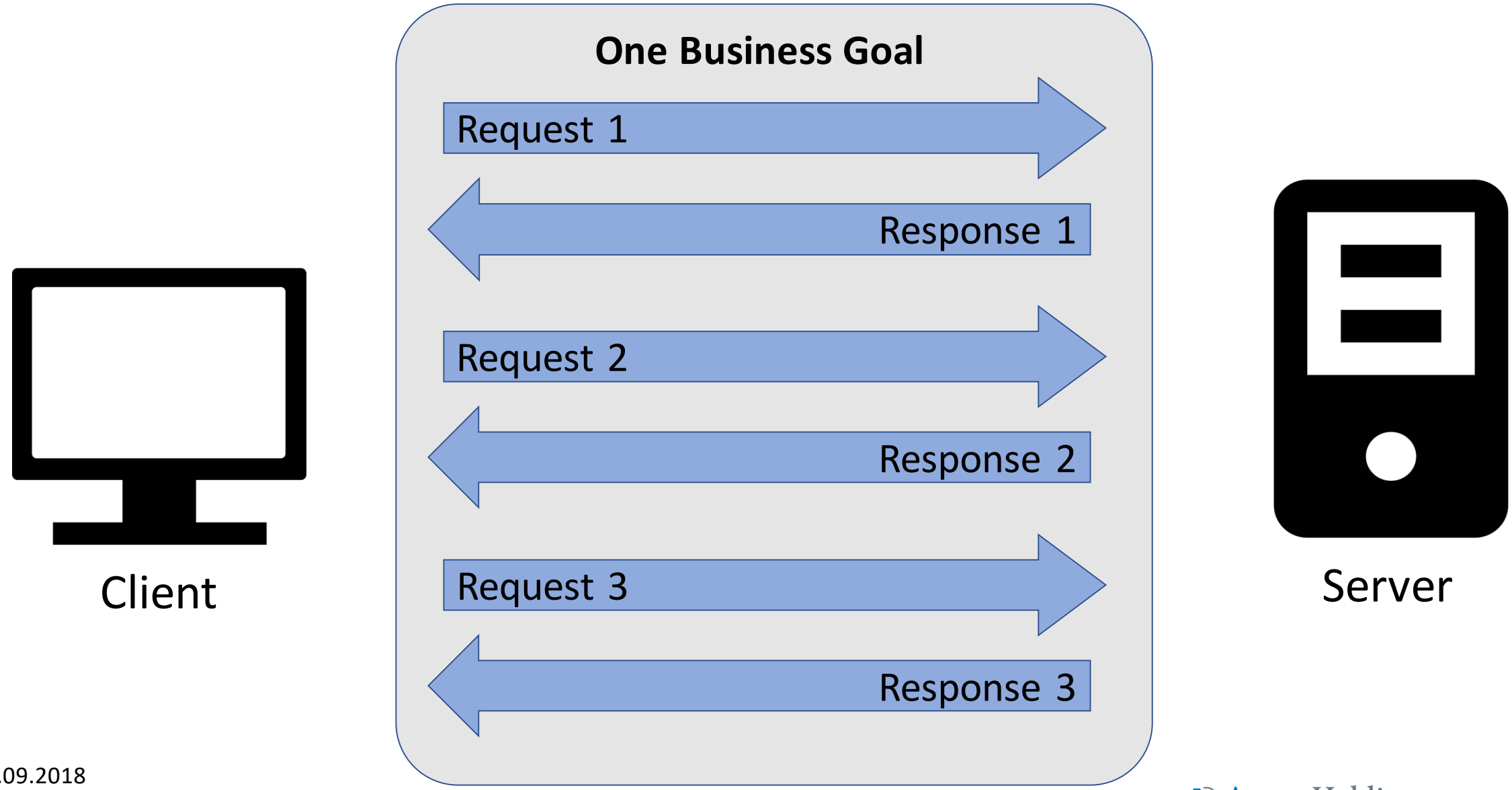
REST - Representational State Transfer

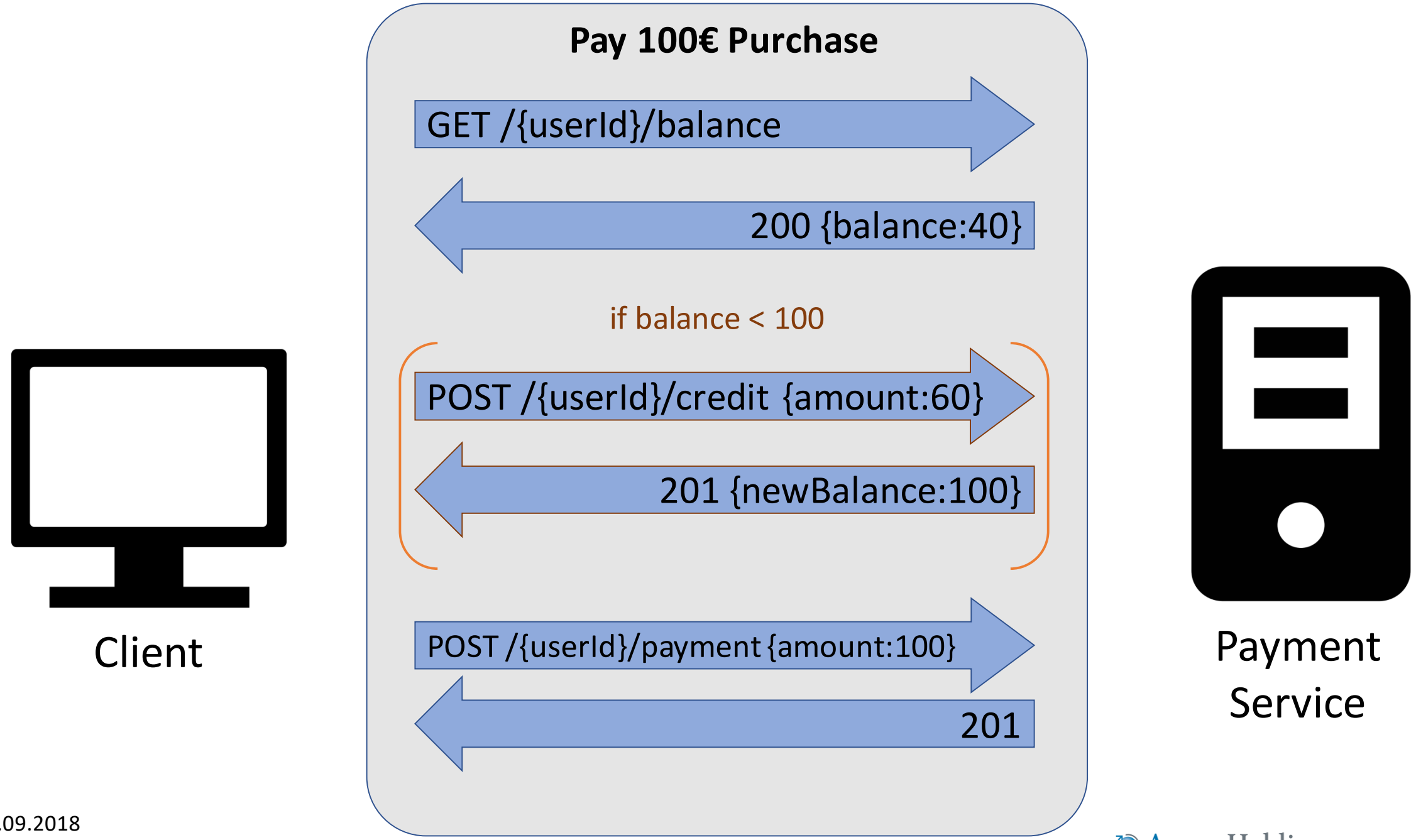


RESTful Processes



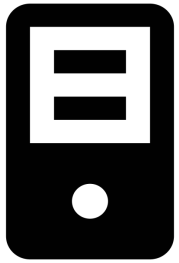
RESTful Processes




Pay 100€ Purchase
GET /{userId}/balance
200 {balance:40}
if balance < 100
POST /{userId}/credit {amount:60}
201 {newBalance:100}
POST /{userId}/payment {amount:100}
201
Client
Payment Service

RESTful Processes

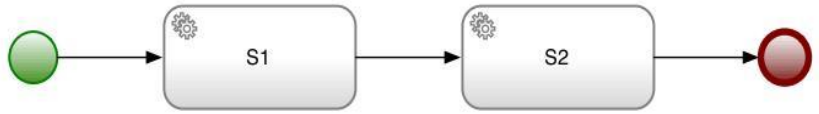
Multiple connect resources, one goal

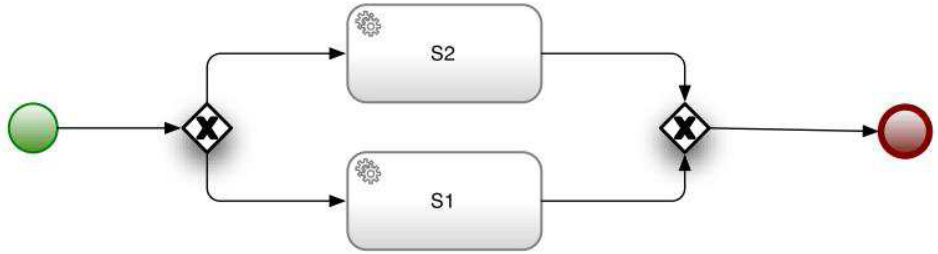


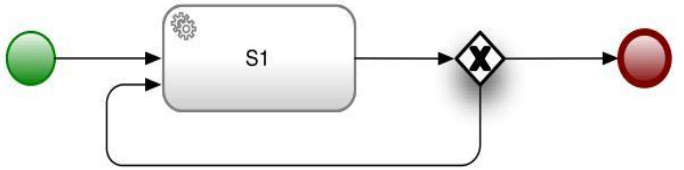
- owns the process
- decides which requests are available

- initiates the process and each request
- must know which request to send next

Control flows

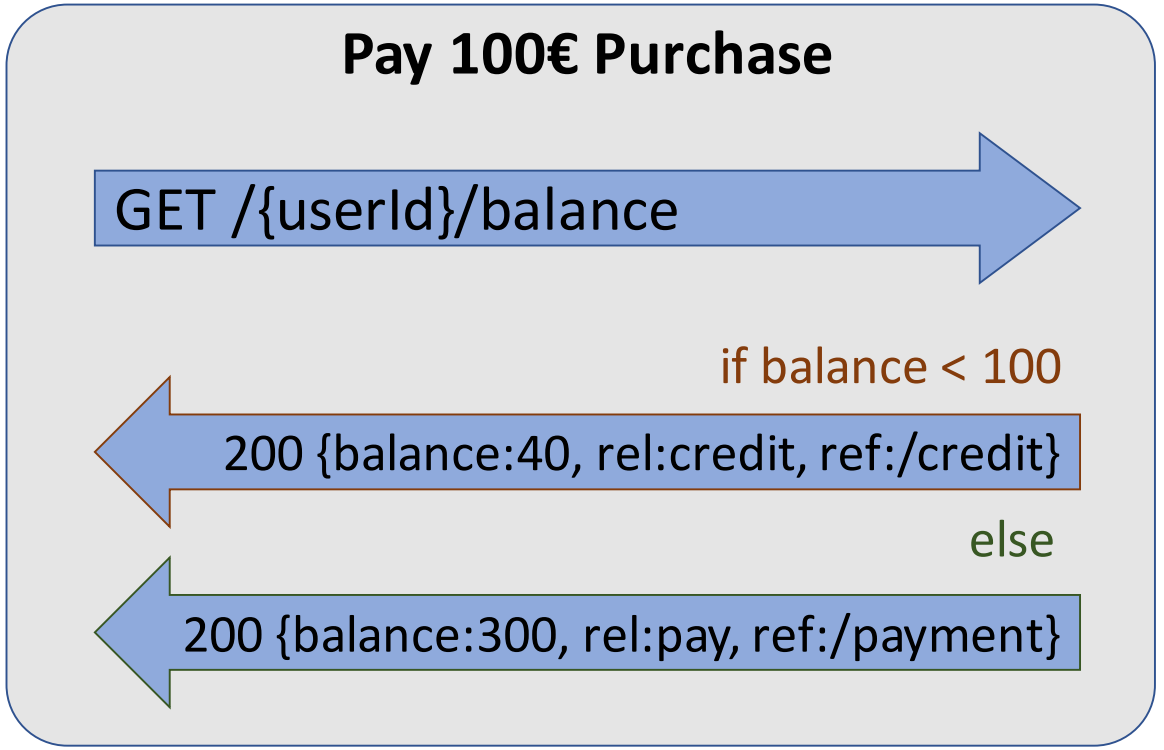




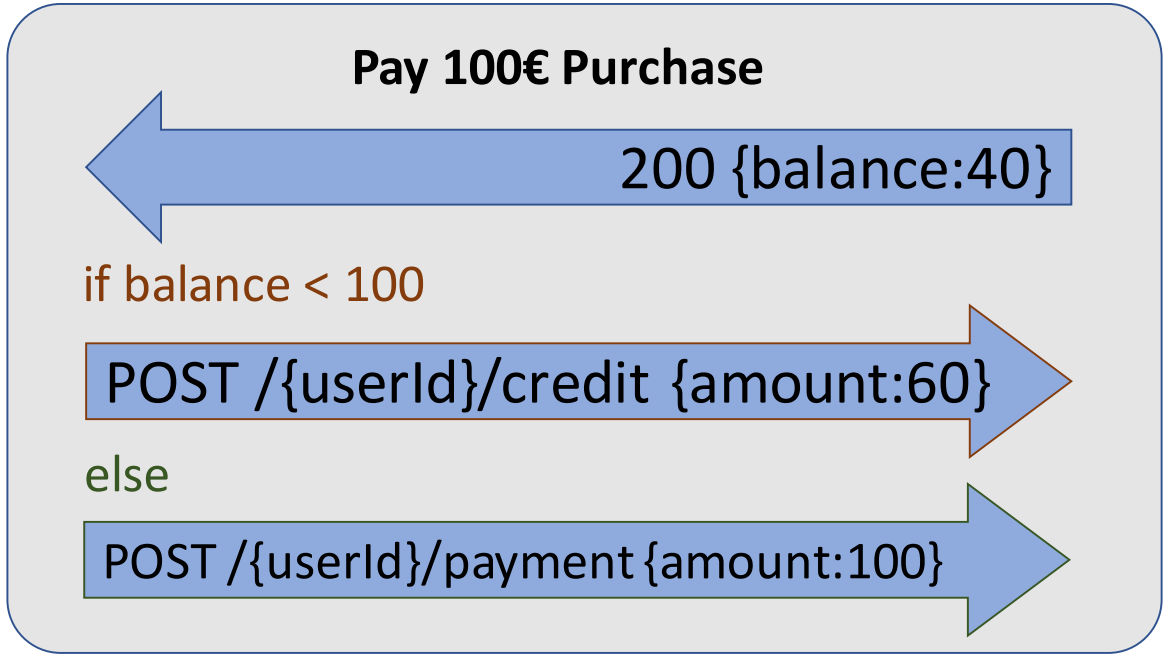


Hypermedia As The Engine Of Application State

- core concept of REST
- server guides client
- decouples client and server
- client must not be aware of control flow
- no need for hardcoded URIs
- increased development effort
- hardly known and implement



RESTful Processes w/o Hypermedia



- server does not guide client
- client needs to decide which requests can be send
- documentation for client developers required

RESTful Processes In Reality

Analysis of 35.000 APIs and two hypermedia APIs

- 50% simple CRUD APIs
- complex APIs implement on average 3 processes and 25 operations
- execution splits based on request/response body
- parallel execution (optional requests)
- OpenAPI Link objects are not used

Capabilities of Swagger/OpenAPI

Allows low level interface (HTTP) documentation

- request (URI, parameter, body)
- response (statuscode, body,
- authentication & authorization

Does not allow high level process documentation

- control flows
- requests dependencies/availability

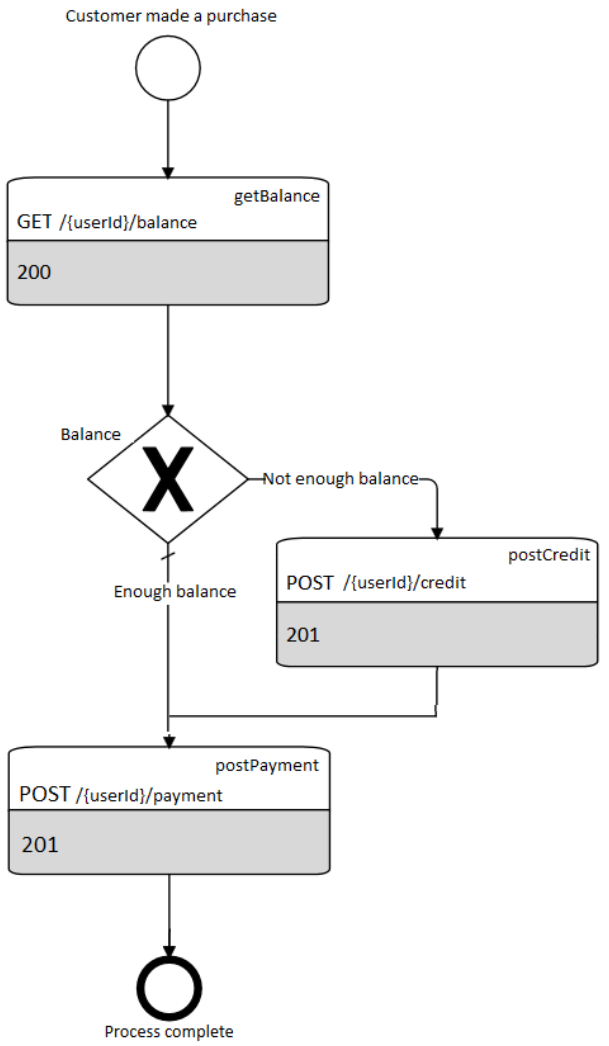
Enabling Process Documentation in OpenAPI

- based on BPMN (gateways, events)
- new schema object to document processes
- does not break existing documentation/tooling
- using existing schema objects/documentation

```yaml
openapi: 3.0.0
info:
  version: 1.0.0
  title: DevCamp 2018
paths:
 '/{userId}/balance':
 '/{userId}/credit':
 '/{userId}/payment':
components:
x-processes:
  paymentProcess:
    summary: Process payment
    activities:
      customerPurchase:
        summary: Customer made a purchase
        event: start
        outgoing:
          linksTo:
            - activity: queryBalance
      queryBalance:
        summary: Query current balance
        operation: getBalance
        outgoing:
          '200':
            conditionParameters:
              - balance
            gateway: exclusive
            linksTo:
              - activity: newCredit
                summary: Not enough balance
                condition: '{$customerPurchase
                  .response.body#/balance}==100'
              - activity: performPayment
                summary: Enough balance
                default: true
      newCredit:
        summary: Open Credit
        operation: postCredit
        outgoing:
          linksTo:
            - activity: performPayment
      performPayment:
        summary: Perform Payment
        operation: postPayment
        outgoing:
          linksTo:
            - activity: end
      end:
        summary: Process complete
        event: end
```

Enabling Process Documentation in OpenAPI

- visualization based on RESTalk (Pautasso et al.)
- integration into SwaggerUI via bpmn-js and converter



What’s next

- implement the visualization
- document real world APIs with the extension
- research
  - if developers can easily write the documentation
  - if documentation and visualization helps with integration of an API
- tooling (e.g. Automatically discover processes based on URI structure)
- propose extension to community