Designing Services with Style

Sébastien Mosser

Lecture #1, 17.09.2018

pictures: sxc.hu

SOA $\Rightarrow$ Services

SOA

⇏ Web Services

Services?

Business vision of the **internal** system

Web Services?

Technological stack used to expose a service through Web standards

Lecture

Service

Labs

Web Service

1 From object to services

2 Interface Engineering

3 Design with style

4 Case study: CréditGénéral

From objects to services

1

Objects

↯

Components

↯

Services

Reduce

coupling

Coupling dimensions

Functional

Location

Temporal

Structural

Business

Technical

Service

(~2000)

Component

(~1990)

Object

(~1970)

Object

Fine-grained

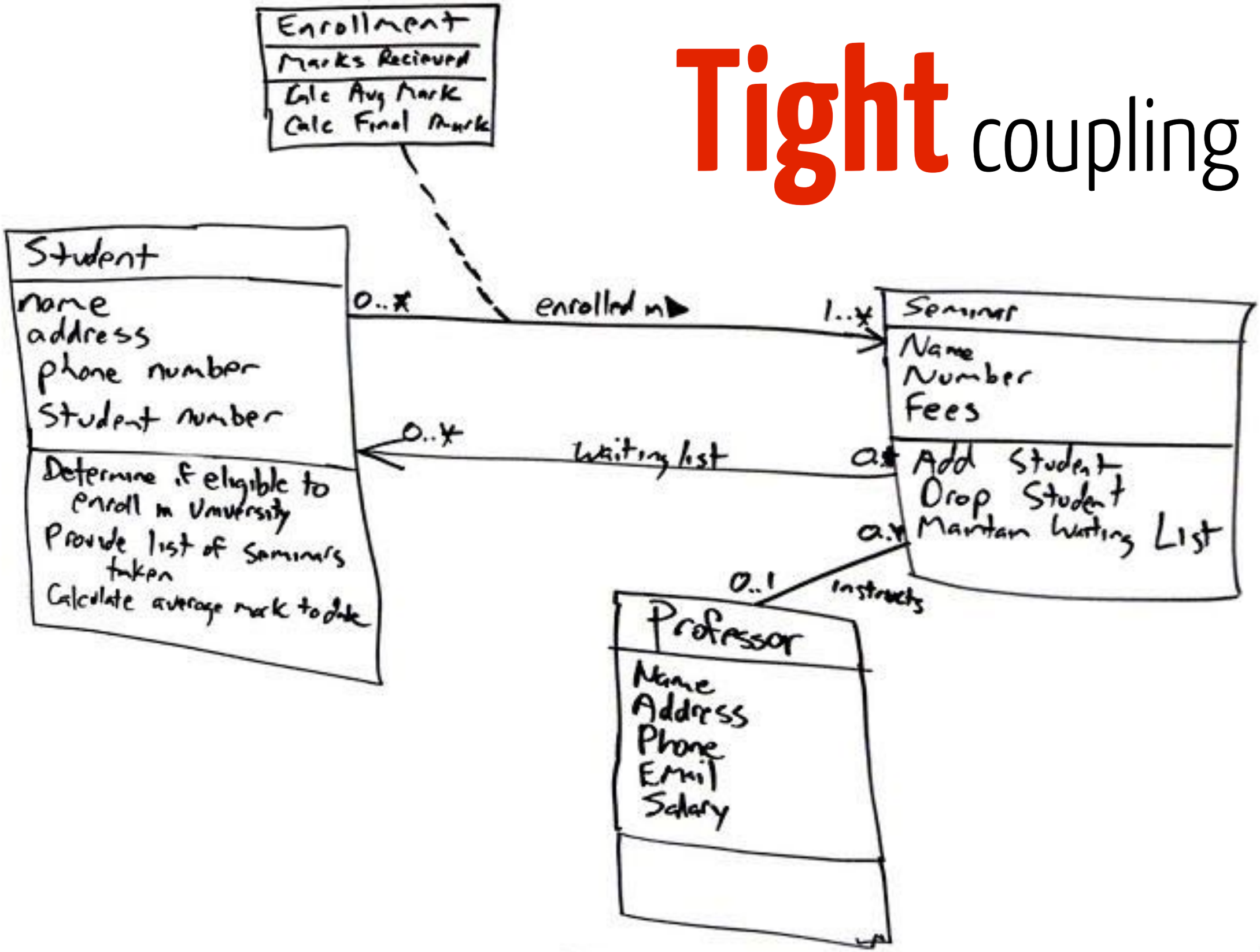
Encapsulate logic ⊕ data

Whitebox reuse

Stateful behavior

Local (often)

Tight coupling



http://www.agilemodeling.com/artifacts/classDiagram.htm

Component*

Coarse-grained

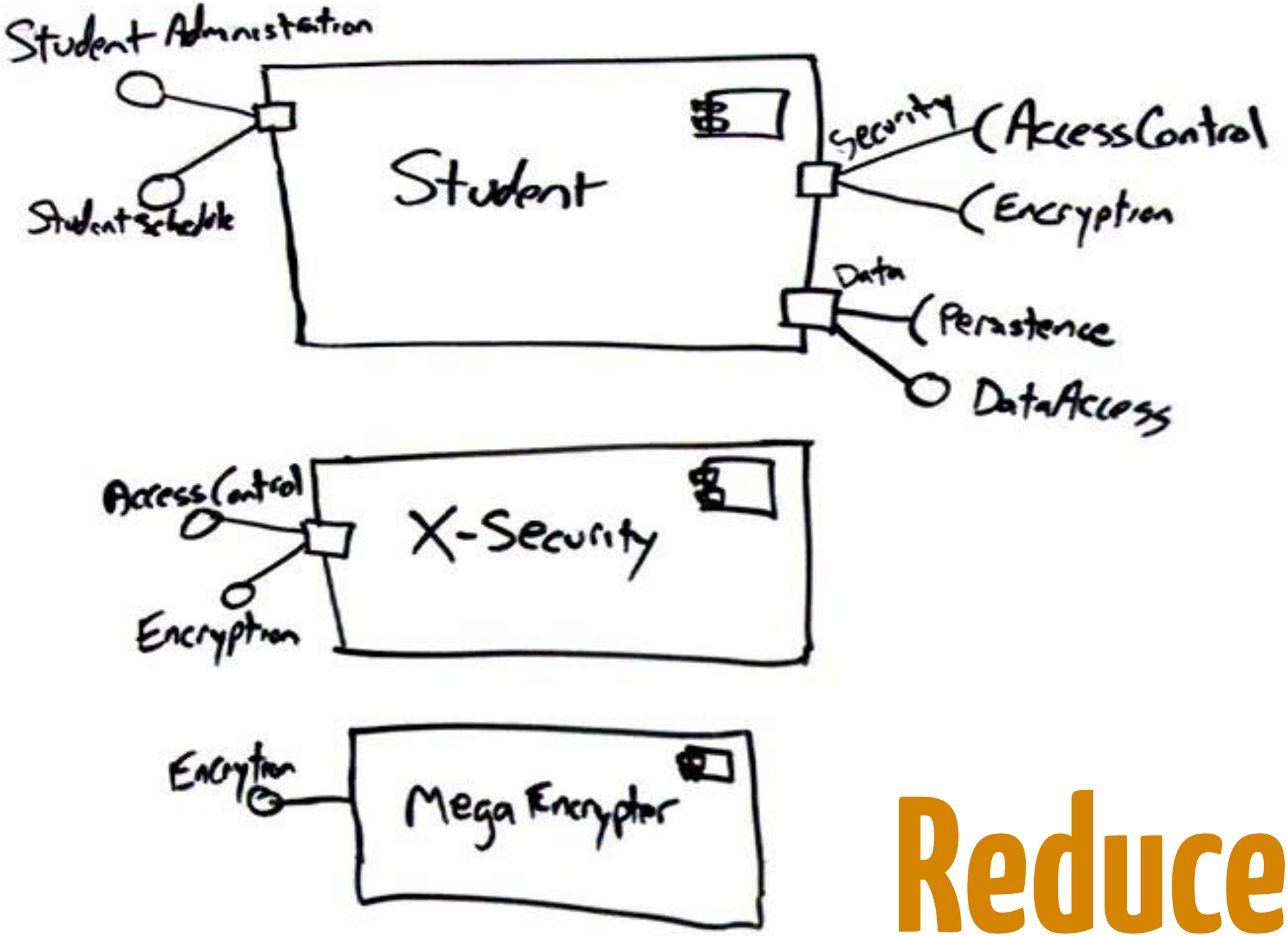
Blackbox reuse

Hosted in a **container**

Behavior **abstraction**

Remote (often)

(*) also valid for distributed objects, EJB, ...



Reduced

coupling

Structural ↘ Functional ↘

http://www.agilemodeling.com/artifacts/componentDiagram.htm

Service

Coarse-grained

Blackbox reuse

Hosted in a **container**

Behaviour **abstraction**

Remote (often)

Fail!

Service = Component ?

C.line (c)

Services

target: customer

are business-oriented

Location ↘
Temporal ↘
Structural ↘ ↘
Functional ↘ ↘

Components

are (more)

code-oriented

target:developer

MicroService

Wait for week #43

But you already have the basics

(and it relies on common sense)

THE EVOLUTION OF

SOFTWARE ARCHITECTURE

1990's

SPAGHETTI-ORIENTED
ARCHITECTURE
(aka Copy & Paste)

2000's

LASAGNA-ORIENTED
ARCHITECTURE
(aka Layered Monolith)

2010's

RAVIOLI-ORIENTED
ARCHITECTURE
(aka Microservices)

WHAT'S NEXT?

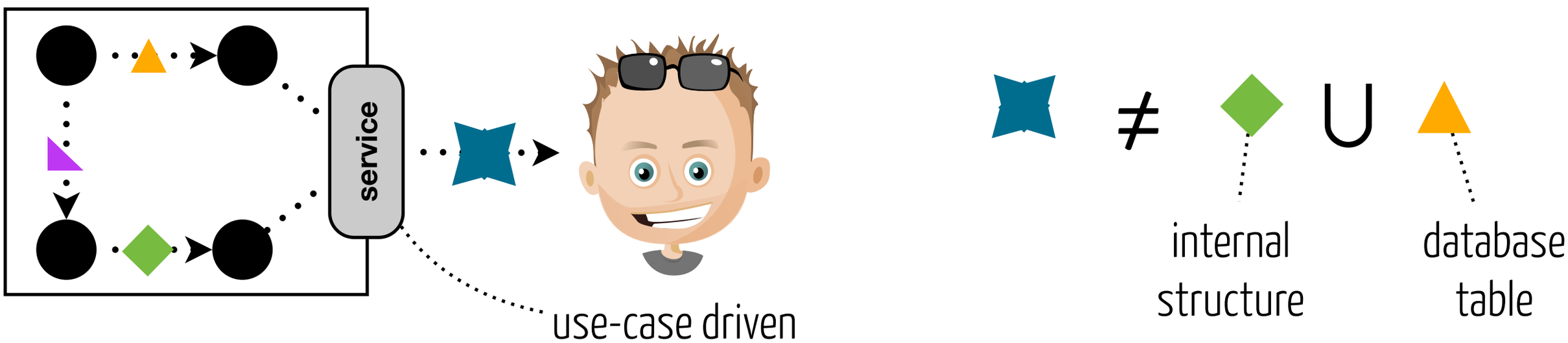
PROBABLY PIZZA-ORIENTED ARCHITECTURE

By @benorama

Interface Engineering

2

Consideration #1: **Encapsulation**



What happens in the system

stays in the system!

Consideration #2: **Service Contract**

«How can I **use** this service?»

«What kind of **request** do you accept?»

«What data structure are **exposed**?»

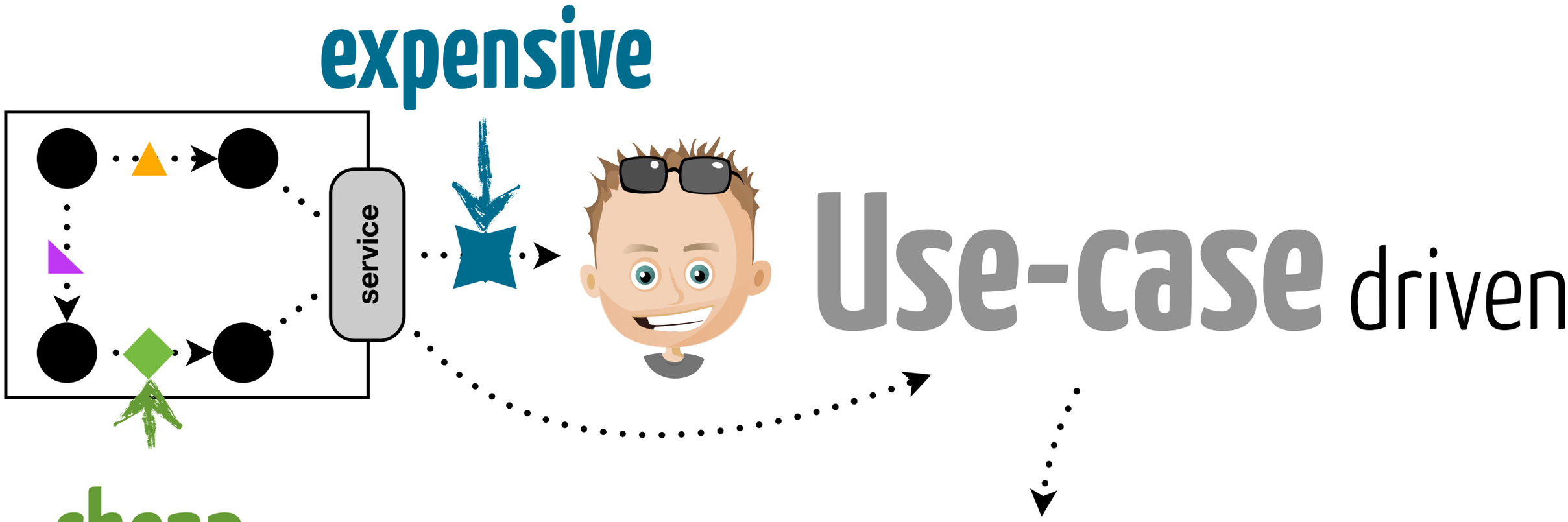
Consideration #3: **Autonomy**

Encourage **stateless** behaviour

Discourage **transactions**

Services **isolate** clients

Consideration #4: **Latency**



Minimize exchanged messages

Networks are **slow**

Consideration #5: **Partial failures**

Networks are **unreliable**

↓

«**Clients** must be **prepared**

for services **to fail**» [SDP]

↙ ↘

Stateless **Idempotency**

Design with style

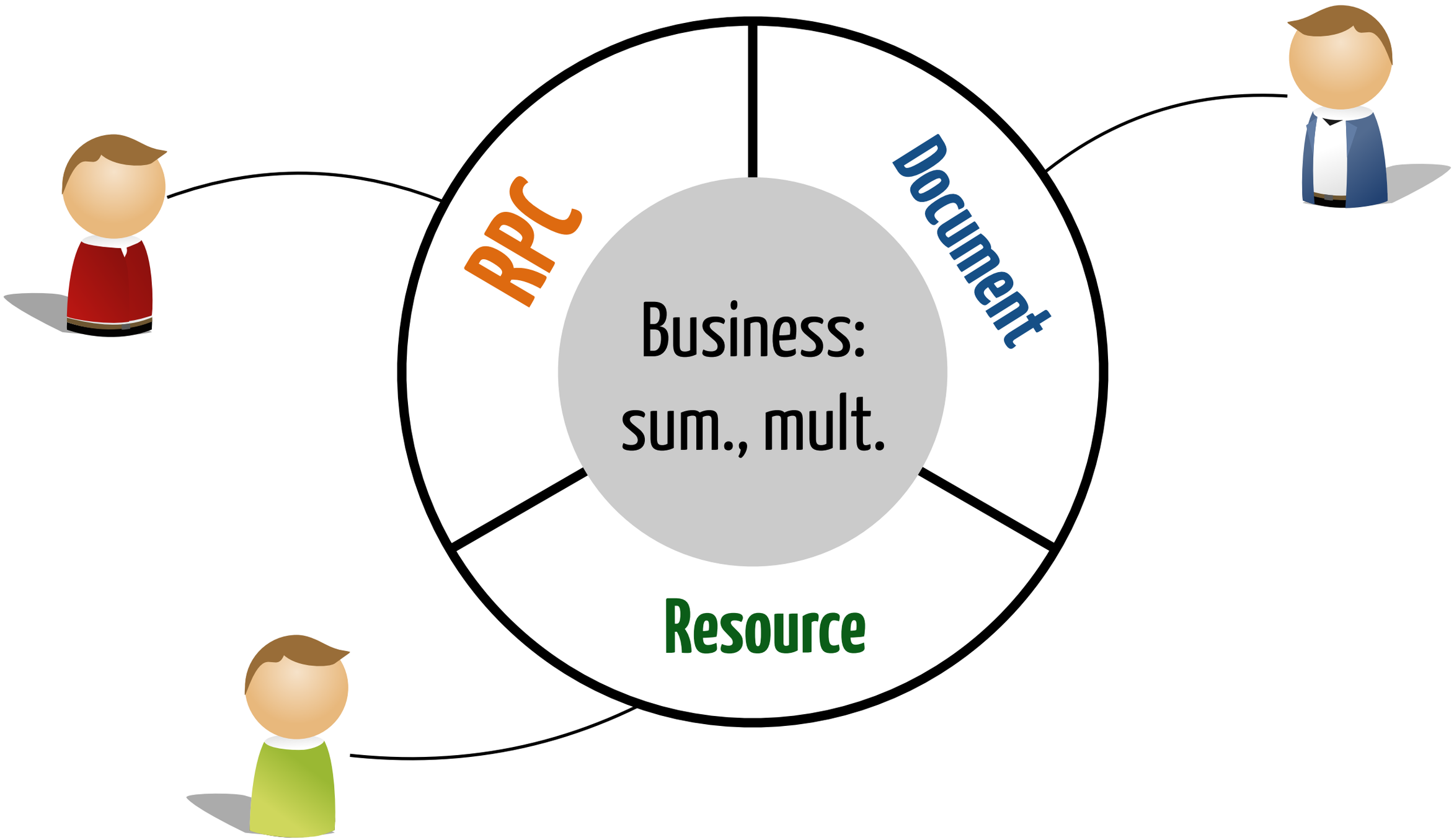
3

Procedure / Document / Resource

Service Interface Styles

Remote Procedure Call	How can clients **execute remote procedures**?
Document (aka Message)	How can clients interact with remote systems while **avoiding direct coupling** to remote procedure?
Resource	How can clients manipulate remote data while **minimizing** the need for **a domain-specific API**?

Example: Calculator

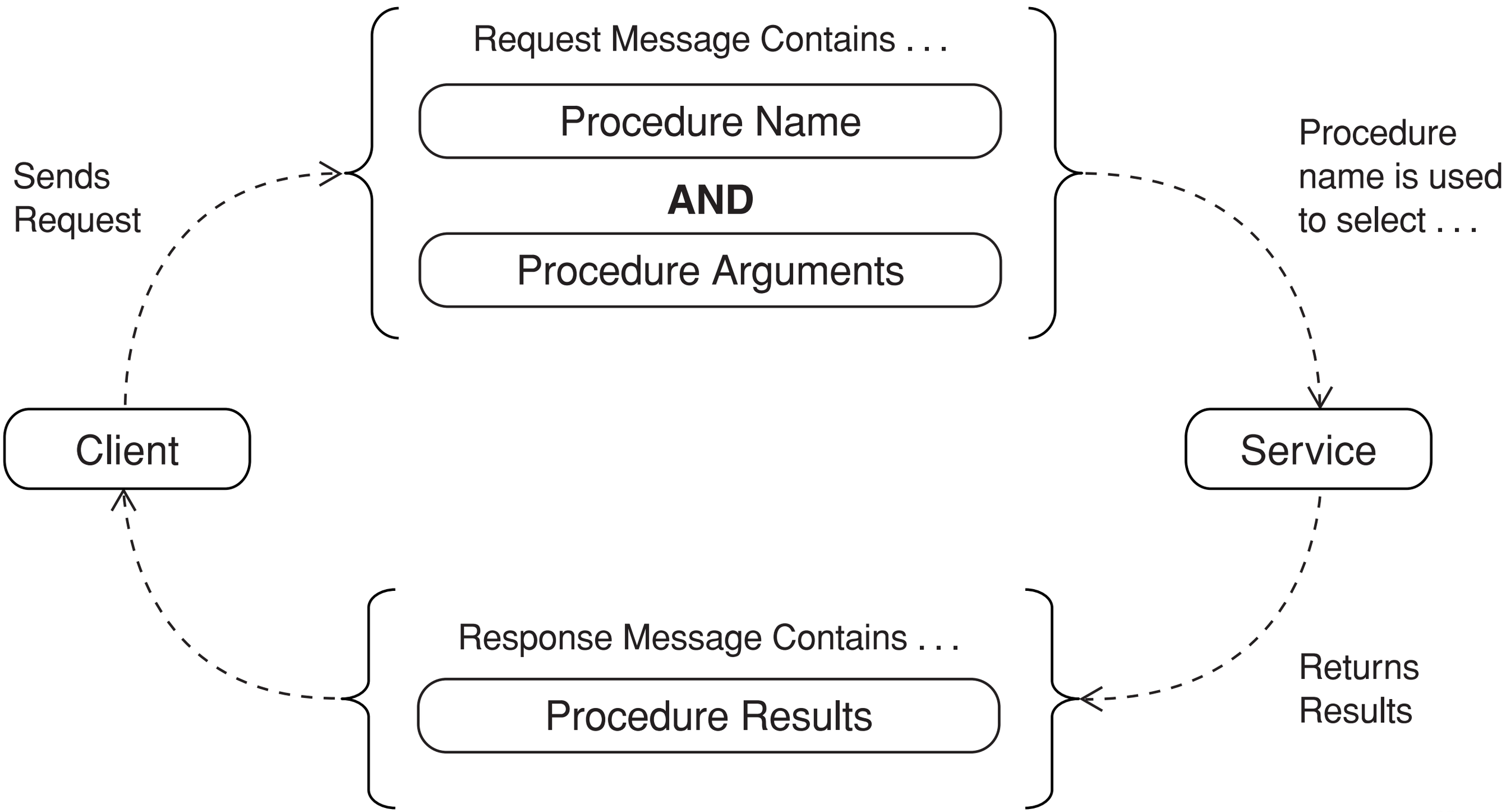


Business (internal) system

```
public class Machine {

   public int addition(int a, int b) {
      return a + b;
   }

   public int multiplication(int a, int b) {
      return a * b;
   }

}
```


https://github.com/polytechnice-si/5A-2014-SOA1/blob/master/webservices/src/main/java/fr/unice/polytech/soa1/calculator/business/Machine.java

RPC Interaction Protocol



[Service Design Patterns]

RPC Interface for "Calculator"

$$\textbf{sumInteger}: \quad \begin{aligned} \text{Int} \times \text{Int} &\longrightarrow \text{Int} \\ (\text{left}, \text{right}) &\longmapsto \text{result} \end{aligned}$$

$$\textbf{multiplyInteger}: \quad \begin{aligned} \text{Int} \times \text{Int} &\longrightarrow \text{Int} \\ (\text{left}, \text{right}) &\longmapsto \text{result} \end{aligned}$$

Implementing an RPC (Web) Service

Code first	Work at the code level Infer the service descriptor Infer the data structures	e.g., Java WSDL XSD
Contract first	Design the data structures Design the service descriptor Generate the code skeletons	XSD WSDL e.g., Java

Consider JSON-RPC as an alternative to SOAP

Calculator: RPC Interface

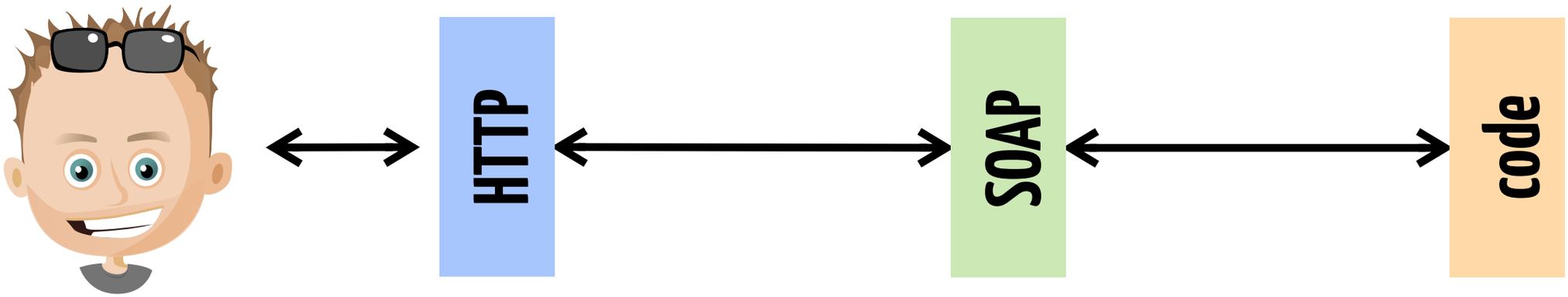
```
@WebService(targetNamespace = "http://…")
public interface CalculatorRPC {

    @WebMethod(operationName = "sumIntegers")
    @WebResult(name = "sum")
    public int sum(  @WebParam(name="left")  int add1,
                     @WebParam(name="right") int add2);

    @WebMethod(operationName = "multiplyIntegers")
    @WebResult(name = "product")
    public int multiply(  @WebParam(name="left")  int mul1,
                          @WebParam(name="right") int mul2);

}
```

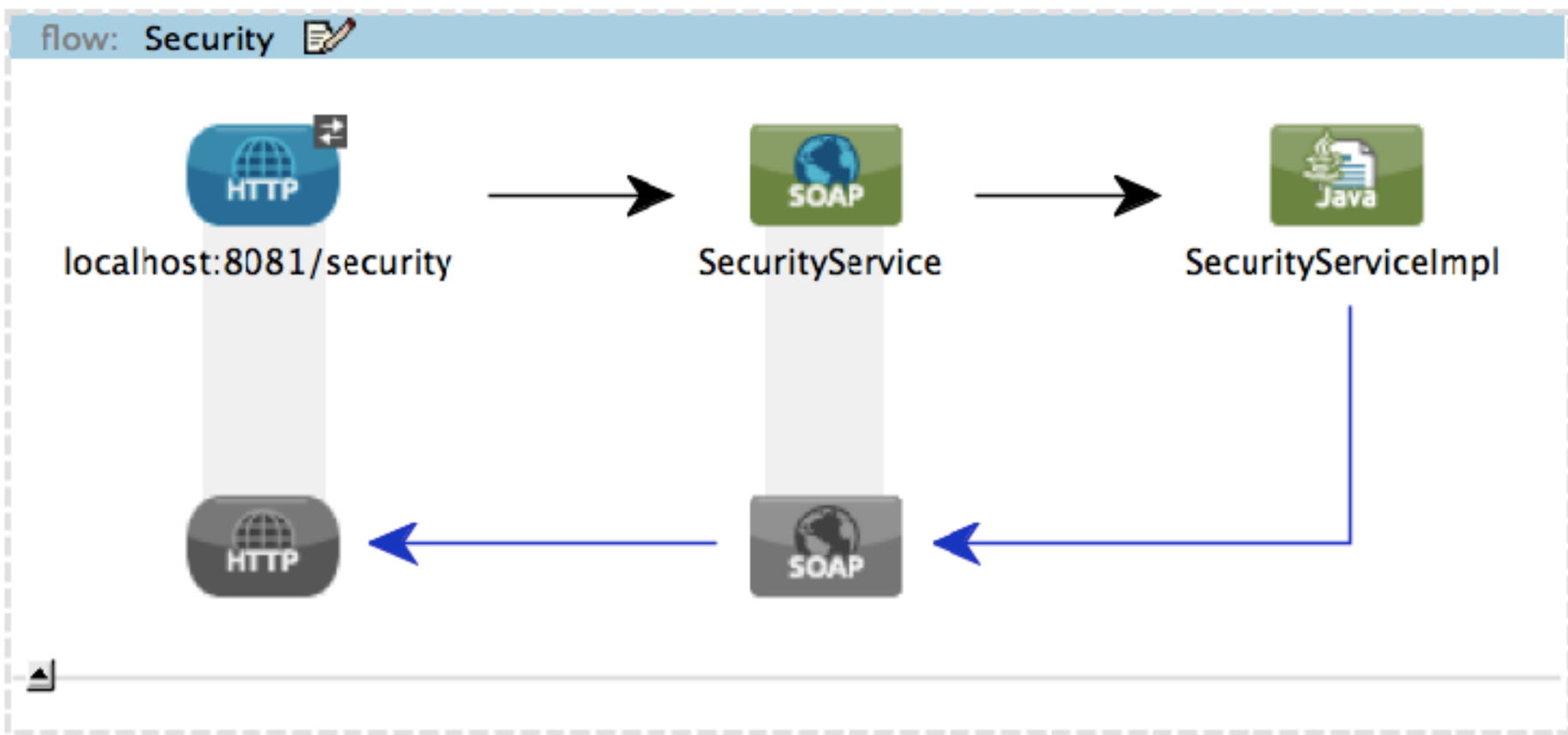

https://github.com/polytechnice-si/5A-2014-SOA1/blob/master/webservices/src/main/java/fr/unice/polytech/soa1/calculator/rpc/CalculatorRPC.java

Calculator: RPC Implementation

```
@Stateless(name = "CalculatorRPC"  )
@WebService(     targetNamespace = "http://…",
                 portName = "CalculatorPort",
                 serviceName = "CalculatorRPCService",
                 endpointInterface = "CalculatorRPC")
public class CalculatorRPCImpl implements CalculatorRPC {

   private Machine calculator = new Machine();

   @Override
   public int sum(int add1, int add2) {
      return calculator.addition(add1, add2);
   }

   @Override
   public int multiply(int mul1, int mul2) {
      return calculator.multiplication(mul1, mul2);
   }
}
```

Exposing the service as a Web Service





WSDL contract to describe it

localhost:8080//webservic

localhost:8080//webservices/CalculatorRPC?wsdl

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:tns="http://informatique.polytech.unice.fr/soa1/calculator/rpc-demo"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:ns1="http://schemas.xmlsoap.org/soap/http"
 name="CalculatorRPCService"
 targetNamespace="http://informatique.polytech.unice.fr/soa1/calculator/rpc-demo">
 ▼<wsdl:types>
  ▶<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:tns="http://informatique.polytech.unice.fr/soa1/calculator/rpc-demo"
   elementFormDefault="unqualified"
   targetNamespace="http://informatique.polytech.unice.fr/soa1/calculator/rpc-demo" version="1.0">...
   </xs:schema>
  </wsdl:types>
 ▼<wsdl:message name="multiplyIntegers">
   <wsdl:part element="tns:multiplyIntegers" name="parameters"></wsdl:part>
  </wsdl:message>
 ▼<wsdl:message name="sumIntegers">
   <wsdl:part element="tns:sumIntegers" name="parameters"></wsdl:part>
  </wsdl:message>
 ▼<wsdl:message name="sumIntegersResponse">
   <wsdl:part element="tns:sumIntegersResponse" name="parameters"></wsdl:part>
  </wsdl:message>
 ▼<wsdl:message name="multiplyIntegersResponse">
   <wsdl:part element="tns:multiplyIntegersResponse" name="parameters"></wsdl:part>
  </wsdl:message>
 ▼<wsdl:portType name="CalculatorRPC">
  ▼<wsdl:operation name="multiplyIntegers">
    <wsdl:input message="tns:multiplyIntegers" name="multiplyIntegers"></wsdl:input>
```

Leveraging WSDL to invoke the service

Request 1

http://localhost:8080/webservices/CalculatorRPC

Raw XML

```
<soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:rpc="http://informatique.polytech.unice.fr/soal/calculator/rpc-demo">
   <soapenv:Header/>
   <soapenv:Body>
      <rpc:sumIntegers>
         <left>3</left>
         <right>2</right>
      </rpc:sumIntegers>
   </soapenv:Body>
</soapenv:Envelope>
```

Auth Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Property (0)

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <ns2:sumIntegersResponse
          xmlns:ns2="http://informatique.polytech.unice.fr/soal/calculator/rpc-demo">
         <sum>5</sum>
      </ns2:sumIntegersResponse>
   </soap:Body>
</soap:Envelope>
```

Pros and **Cons** of RPC Services

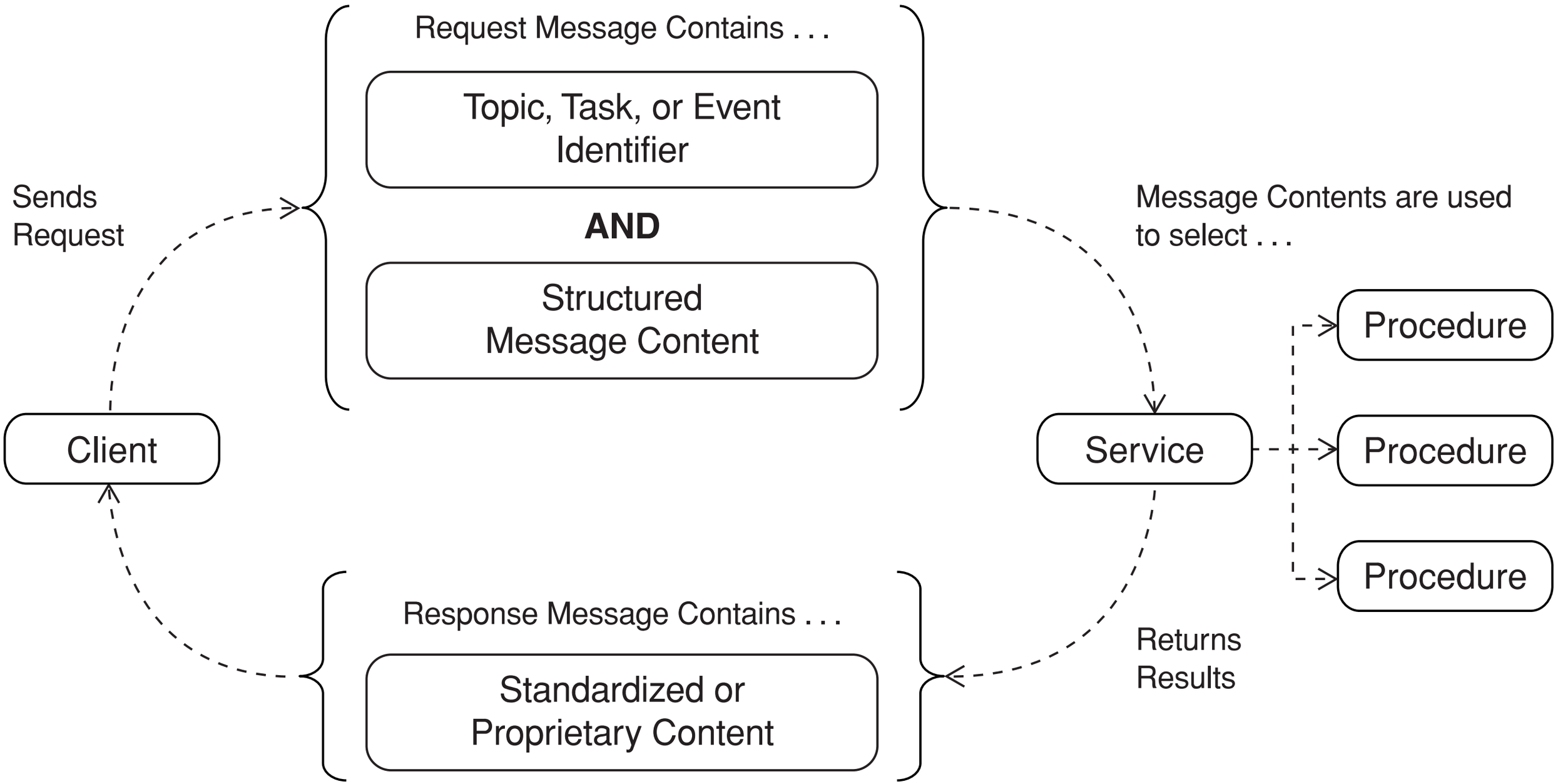
As **easy** as A-B-C

Flat Interfaces «by nature»

Encapsulate remote invocation

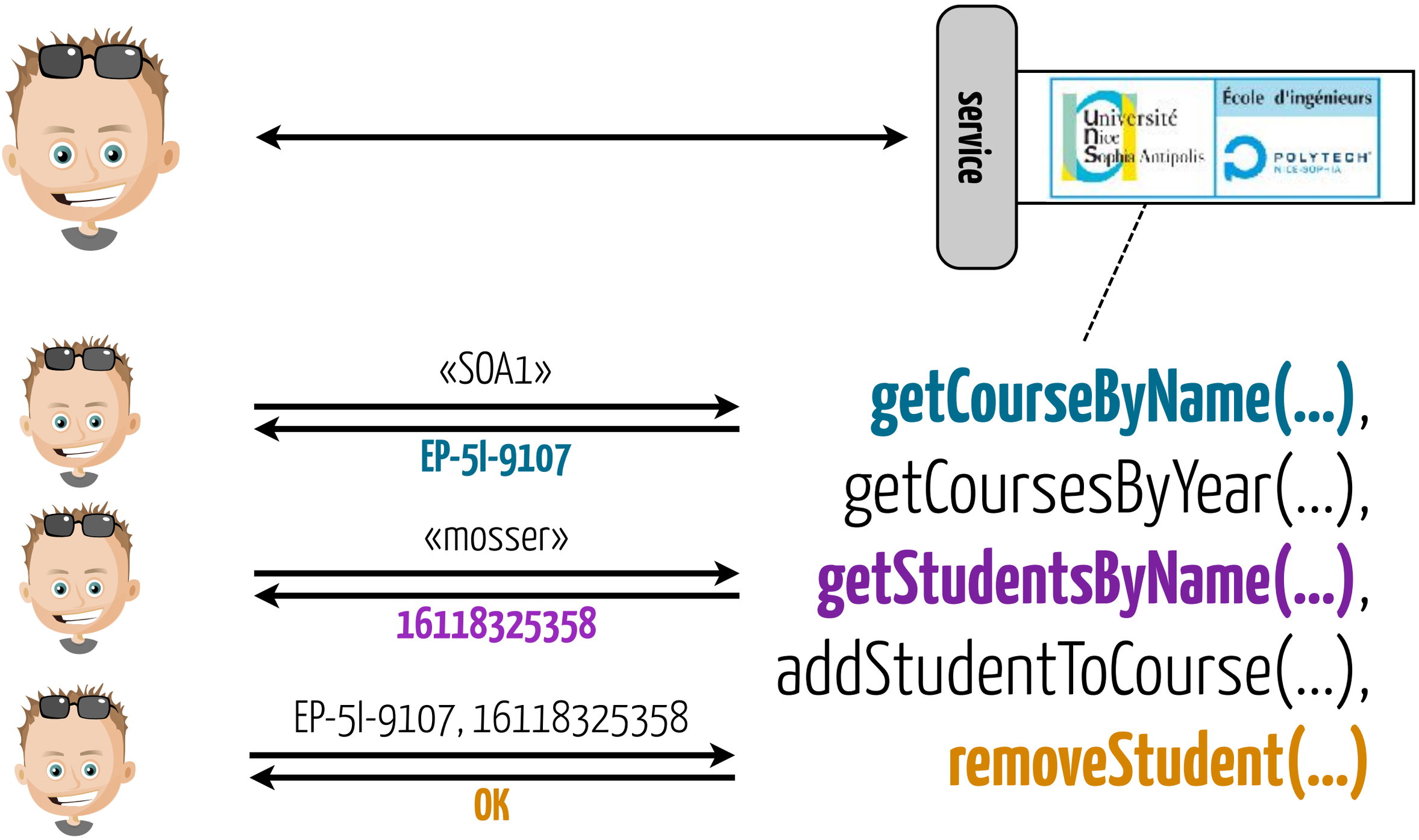
Tightly coupled to procedures

Document Interaction Protocol

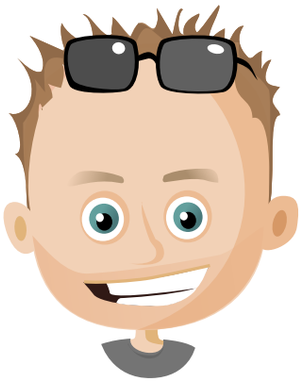


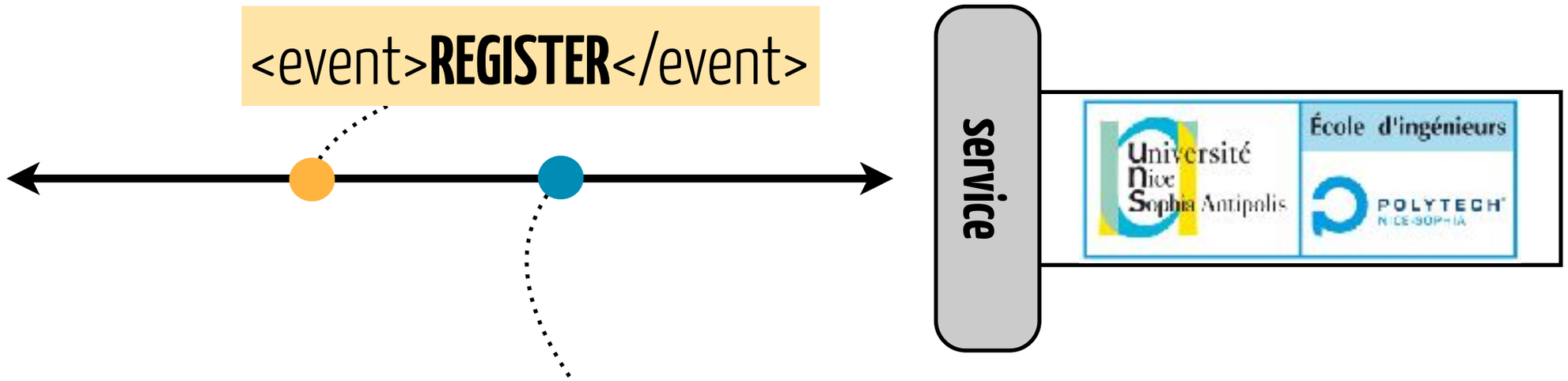
«As a student, I decide to **give up** the **SOA1** course.»

RPC Approach



What about …



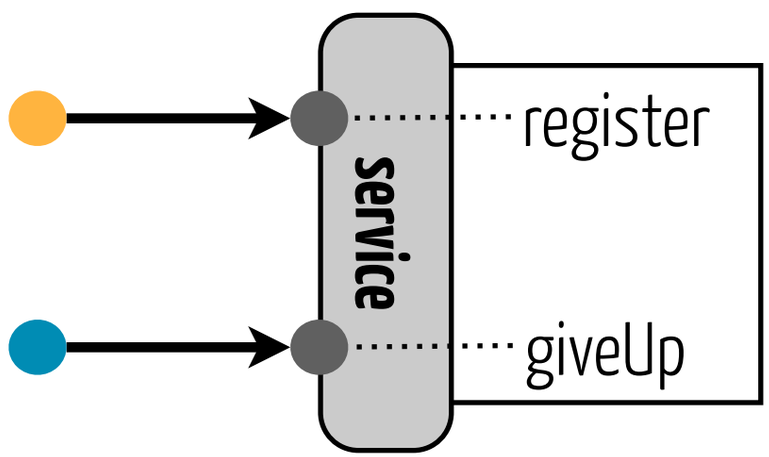


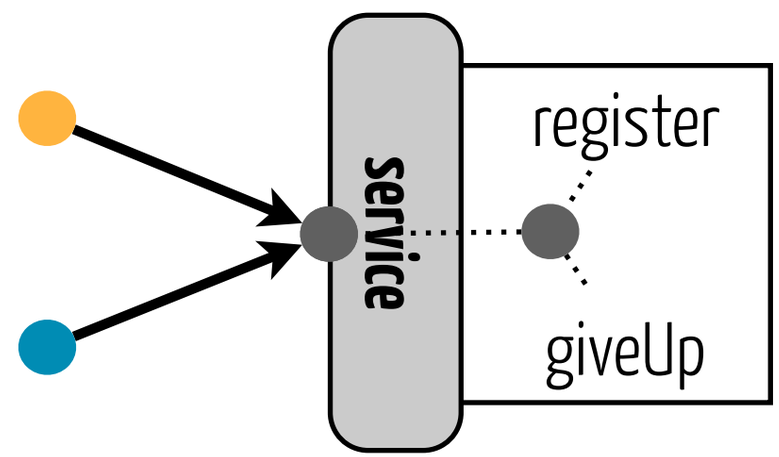
«As a student, **I** decide to **give up** the **SOA1** course.»

```
<message>
 <event>GIVE_UP</event>
 <student id="16118325358">
  <first>Sébastien</first><last>MOSSER</last>
 <student>
 <course>
  <dept>SI</dept><year>5</year><acro>SOA1</acro>
 </course>
</message>
```

Message Interface are **built like RPC** ones.

Excepting the procedure **coupling**





Document Interface for "Calculator"

execute: $\text{Input} \longrightarrow \text{Output}$

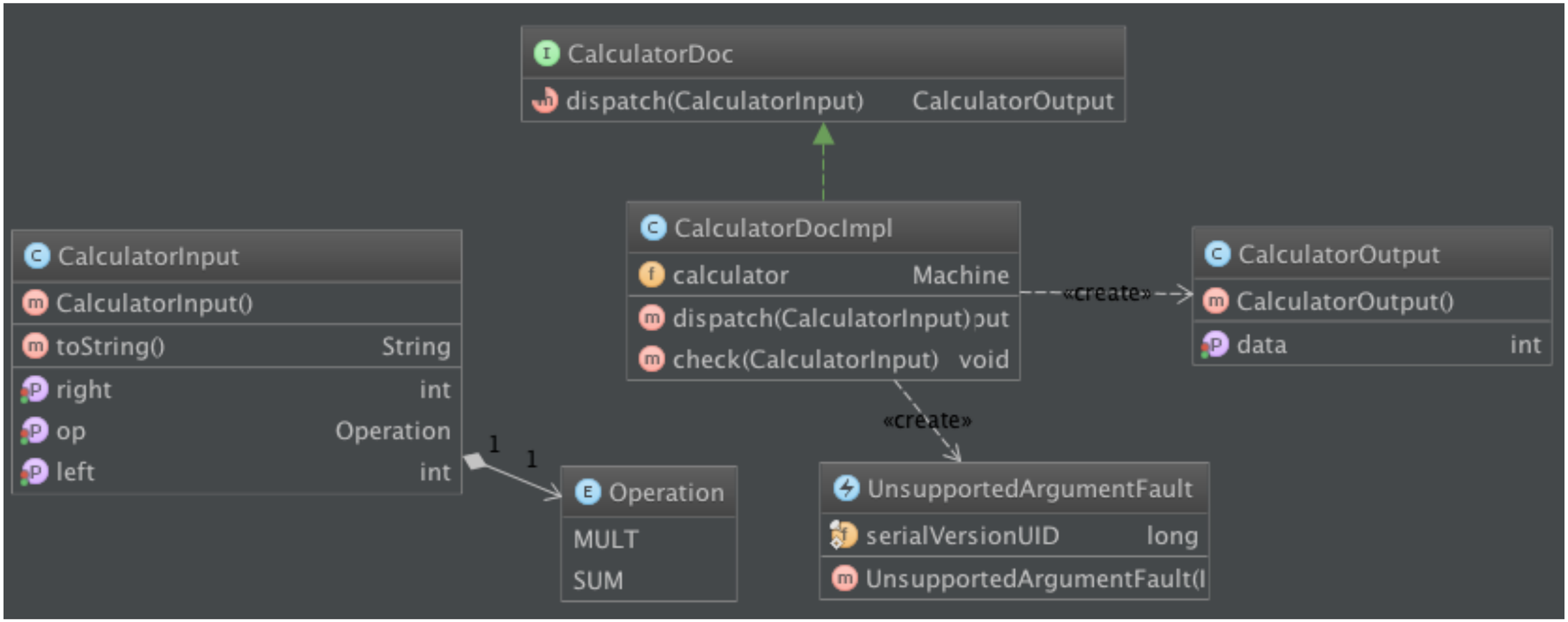
$\text{in} \longmapsto \text{out}$

$$\text{Input} = \text{Int} \times \text{Operator} \times \text{Int}$$

$$\text{Output} = \text{Int}$$

$$\text{Operator} = \{\text{ADD}, \text{MULT}\}$$

Java Implementation



https://github.com/polytechnice-si/5A-2014-SOA1/tree/master/webservices/src/main/java/fr/unice/polytech/soa1/calculator/doc

Calculator Document Interface

```
@WebService(targetNamespace = "http://…")
public interface CalculatorDoc {

    @WebMethod(operationName = "execute")
    @WebResult(name = "processed")
    public CalculatorOutput dispatch(
            @WebParam(name = "job") CalculatorInput input)
        throws UnsupportedArgumentFault;

}
```


https://github.com/polytechnice-si/5A-2014-SOA1/blob/master/webservices/src/main/java/fr/unice/polytech/soa1/calculator/doc/CalculatorDoc.java

Service Implementation

```
@Stateless(name = "CalculatorDOC")
@WebService(…)
public class CalculatorDocImpl implements CalculatorDoc {
    private Machine m = new Machine();
    @Override
    public CalculatorOutput dispatch(CalculatorInput input)
                throws UnsupportedArgumentFault {
        check(input); // only accept positive or null integers
        CalculatorOutput res = new CalculatorOutput();
        switch(input.getOp()) {
            case MULT:
                res.setData(m.multiplication(input.getLeft(), input.getRight()));
                break;
            case ADD:
                res.setData(m.addition(input.getLeft(), input.getRight()));
                break;
        }
        return res;
    }

    private void check(CalculatorInput input) throws UnsupportedArgumentFault {
        // Check if the input message is compliant with internal rules.
    }
}
```


https://github.com/polytechnice-si/5A-2014-SOA1/blob/master/webservices/src/main/java/fr/unice/polytech/soa1/calculator/doc/CalculatorDocImpl.java

Pros and Cons of Message Interfaces

Delegation pattern

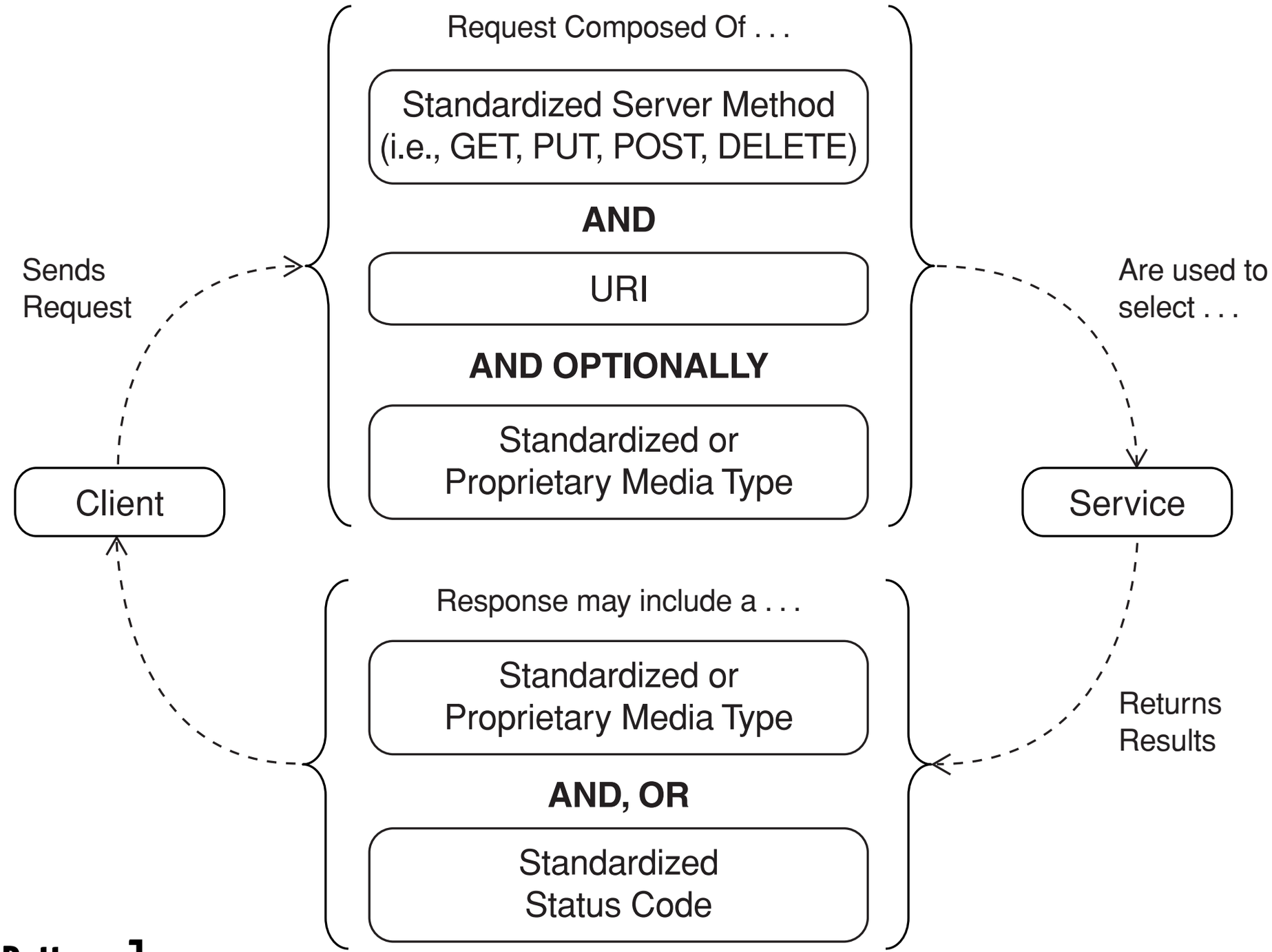
Service Isolation

Weak descriptor

Cumbersome implementation

Business modelled as messages

Resource Interaction Protocol



Avoid

message proliferation

Invert your **point of view**!

From **actions** to **resources**

verbs → **nouns**

Uniform Interface, e.g., **Requests** as HTTP verbs

OPTIONS

HEAD

GET

POST

PUT

DELETE

TRACE

CONNECT

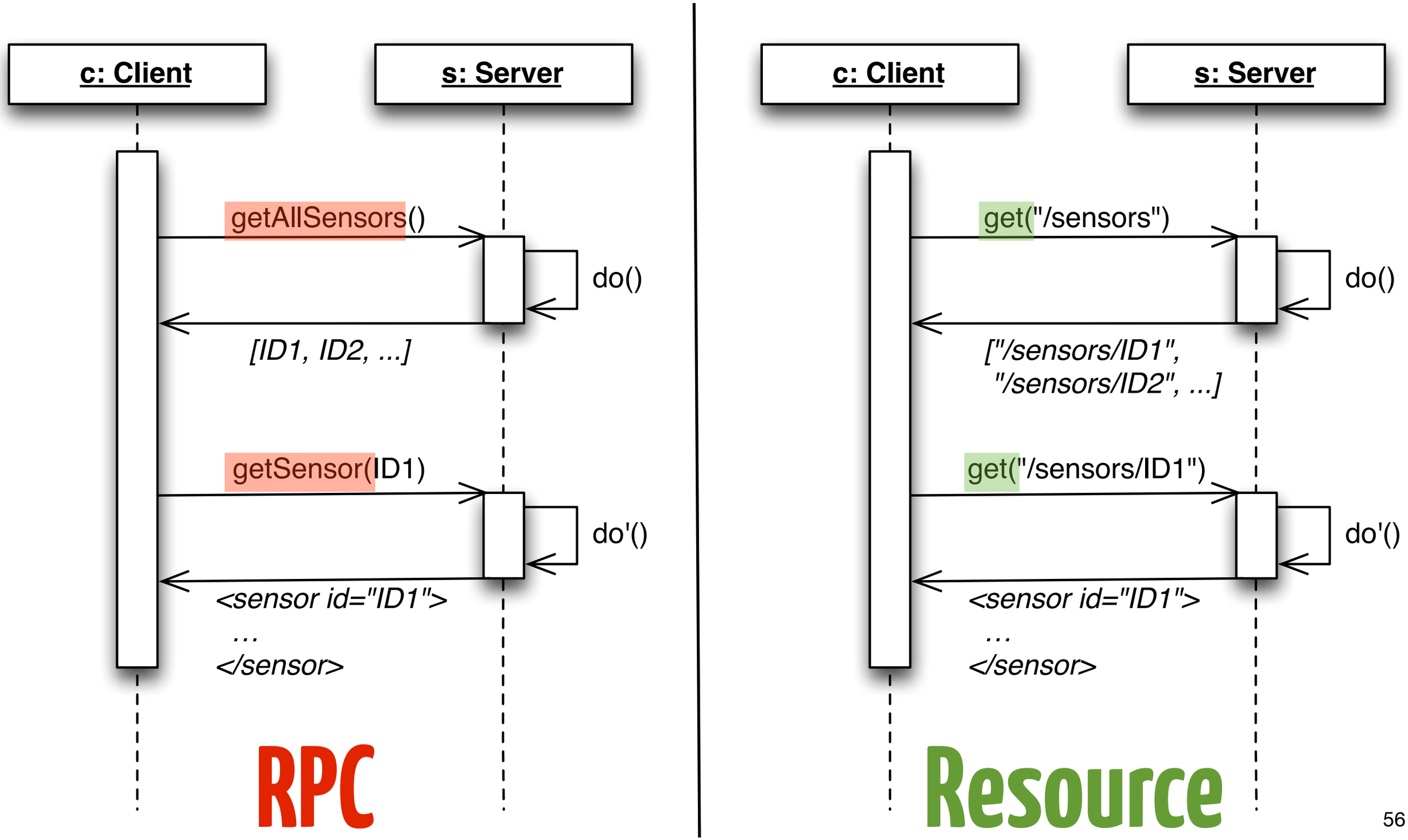
http://www.ietf.org/rfc/rfc2616.txt

Uniform Interface, e.g., **Responses** as HTTP Status

Family	Intention
1xx	Informational
2xx	Successful
3xx	Redirection
4xx	Client Error
5xx	Server Error

http://www.ietf.org/rfc/rfc2616.txt

Resources versus RPC



Resource Interface for "Calculator"

adder: **GET** .../adder/{left}/{right}

multiplier: **POST**(left, right) .../multiplier

Calculator Resource Interface

```
@Produces({"text/plain"})
public interface CalculatorREST {

    @Path("/adder/{left}/{right}")
    @GET
    public int adder( @PathParam("left")   int a,
                      @PathParam("right")  int b);

    @Path("/multiplier")
    @POST
    public int multiplier( @QueryParam("left")  int a,
                           @QueryParam("right") int b);

}
```


https://github.com/polytechnice-si/5A-2014-SOA1/blob/master/webservices/src/main/java/fr/unice/polytech/soa1/calculator/rest/CalculatorREST.java

Calculator Resource Implementation

```
@Path("/rest/calculator")
public class CalculatorRESTImpl implements CalculatorREST {

   private Machine calculator = new Machine();

   @Override
   public int adder(int a, int b) {
     return calculator.addition(a,b);
   }

   @Override
   public int multiplier(int a, int b) {
      return calculator.multiplication(a,b);
   }
}
```


https://github.com/polytechnice-si/5A-2014-SOA1/blob/master/webservices/src/main/java/fr/unice/polytech/soa1/calculator/rest/CalculatorRESTImpl.java

Choosing the right verb

HTTP Verb	/customers	/customers/{id}
GET	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	404 (Not Found), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
POST	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found).
DELETE	404 (Not Found), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

Case Study 4

CréditGénéral

Needs: **CréditGénéral** Payment Services

retailers

commercial agents

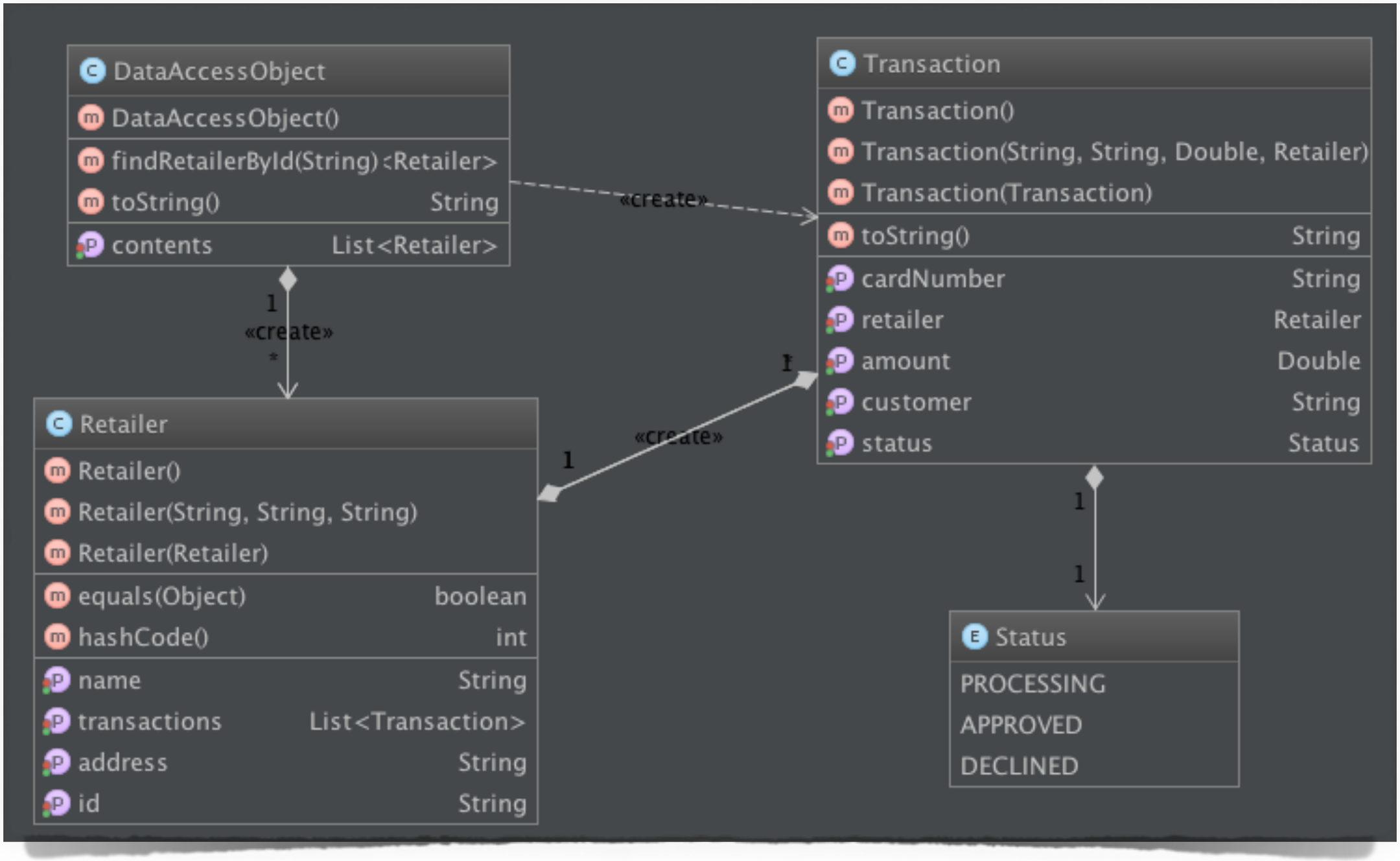
process a card payment

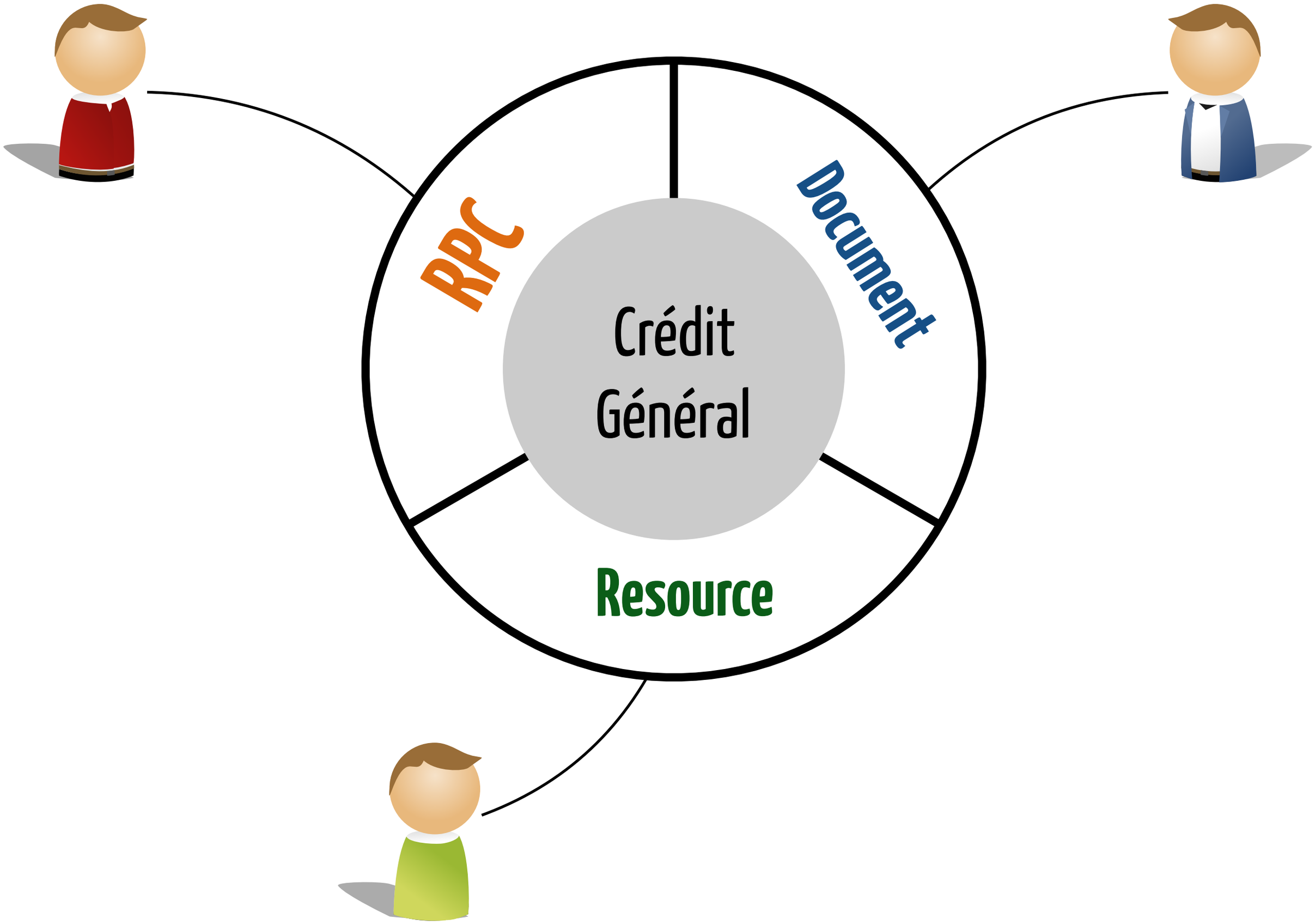
list existing transactions

register new retailers

manage existing retailers

Business Objects




RPC
Document
Crédit Général
Resource

Exercice!

Groups of 4 students

Design an interface for the Payment Service

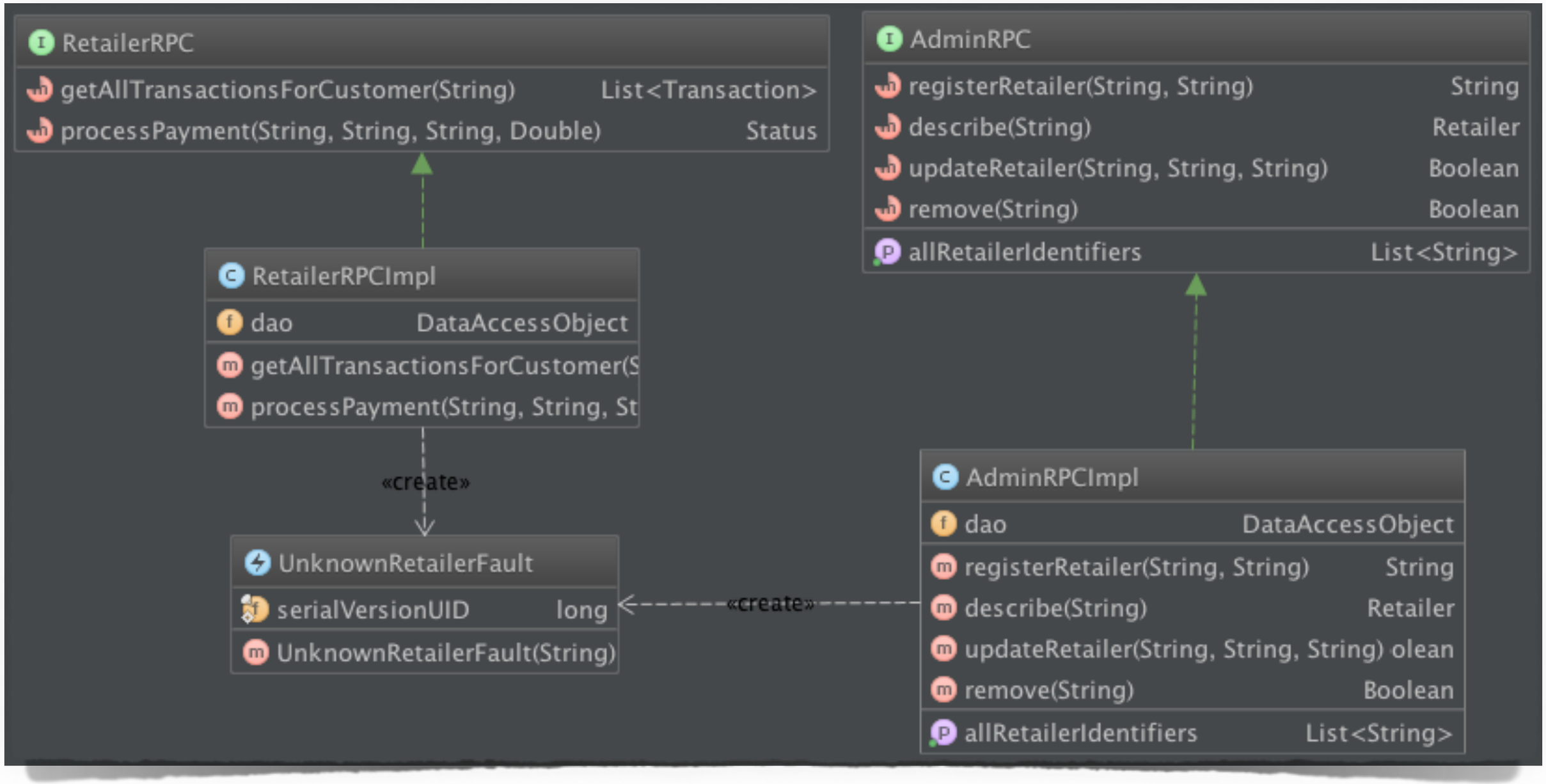
#Procedure ≈ #Document ≈ #Resource

20 minutes

RPC Interface: Proposition

- **Public service**:
  - **get_all_transactions_for_customer**: return all the transactions associated to the retailer given as parameter
  - **process_payment**: process a payment for a given retailer
- **Private service**:
  - **describe_customer**: returns the information stored for an existing retailer
  - **get_all_customer_identifiers**: return all the identifier stored in the system
  - **register_new_retailer**: create a new retailer in the system
  - **remove_customer**: delete an existing retailer from the storage tiers
  - **update_customer**: update the information (i.e., name and address) for an existing retailer.

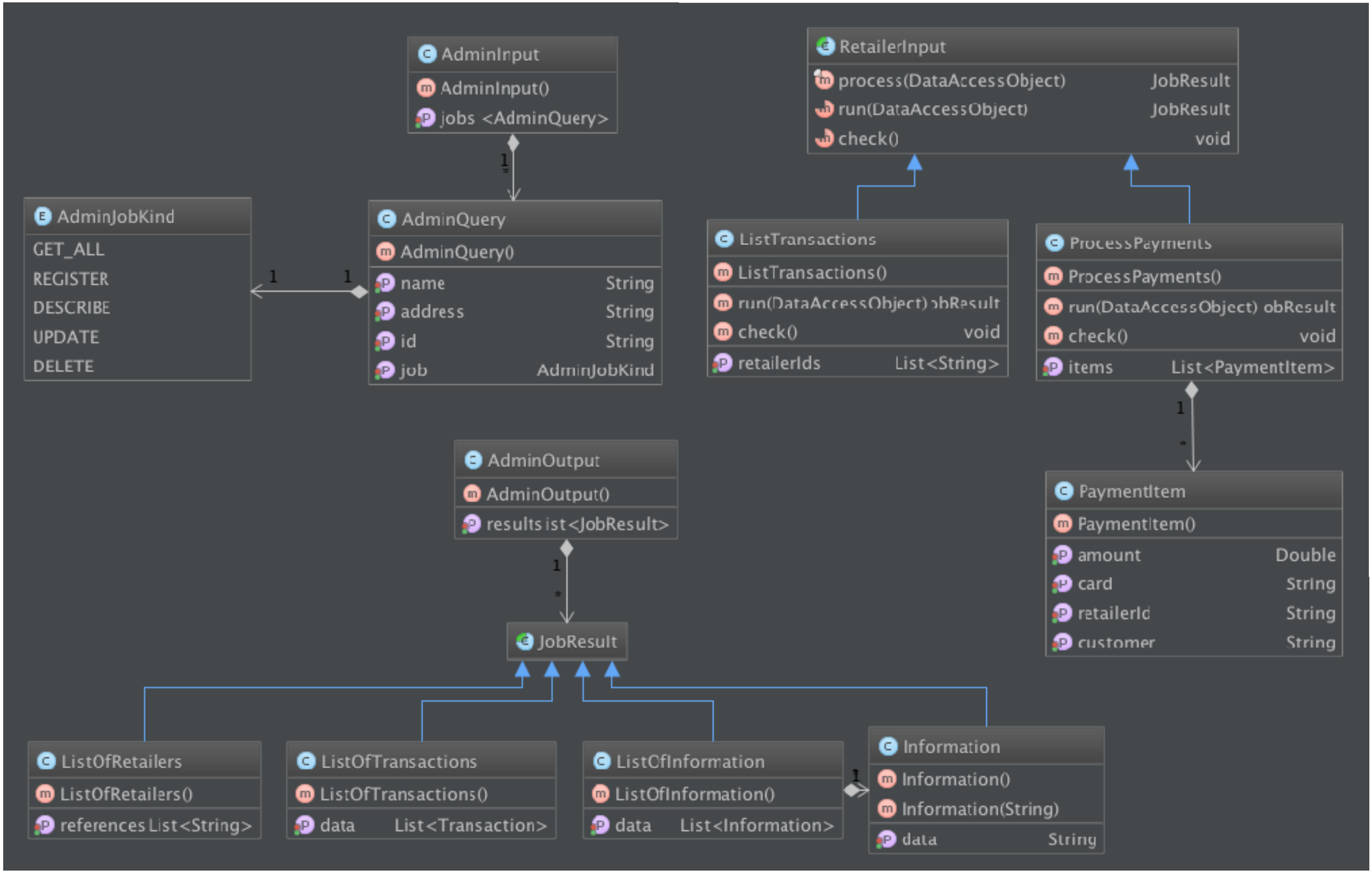
RPC Interface: Implementation



https://github.com/polytechnice-si/5A-2014-SOA1/tree/master/webservices/src/main/java/fr/unice/polytech/soa1/payment/rpc

Document Interface: Proposition

- **Public service:**
  - Process batch of payments (e.g., all payments of the day)
  - Compute sets of transactions (e.g., aggregating local retailers)
- **Private service:**
  - Process a list of job to be executed sequentially
  - Available jobs: GET_ALL, DESCRIBE, REGISTER, UPDATE, DELETE



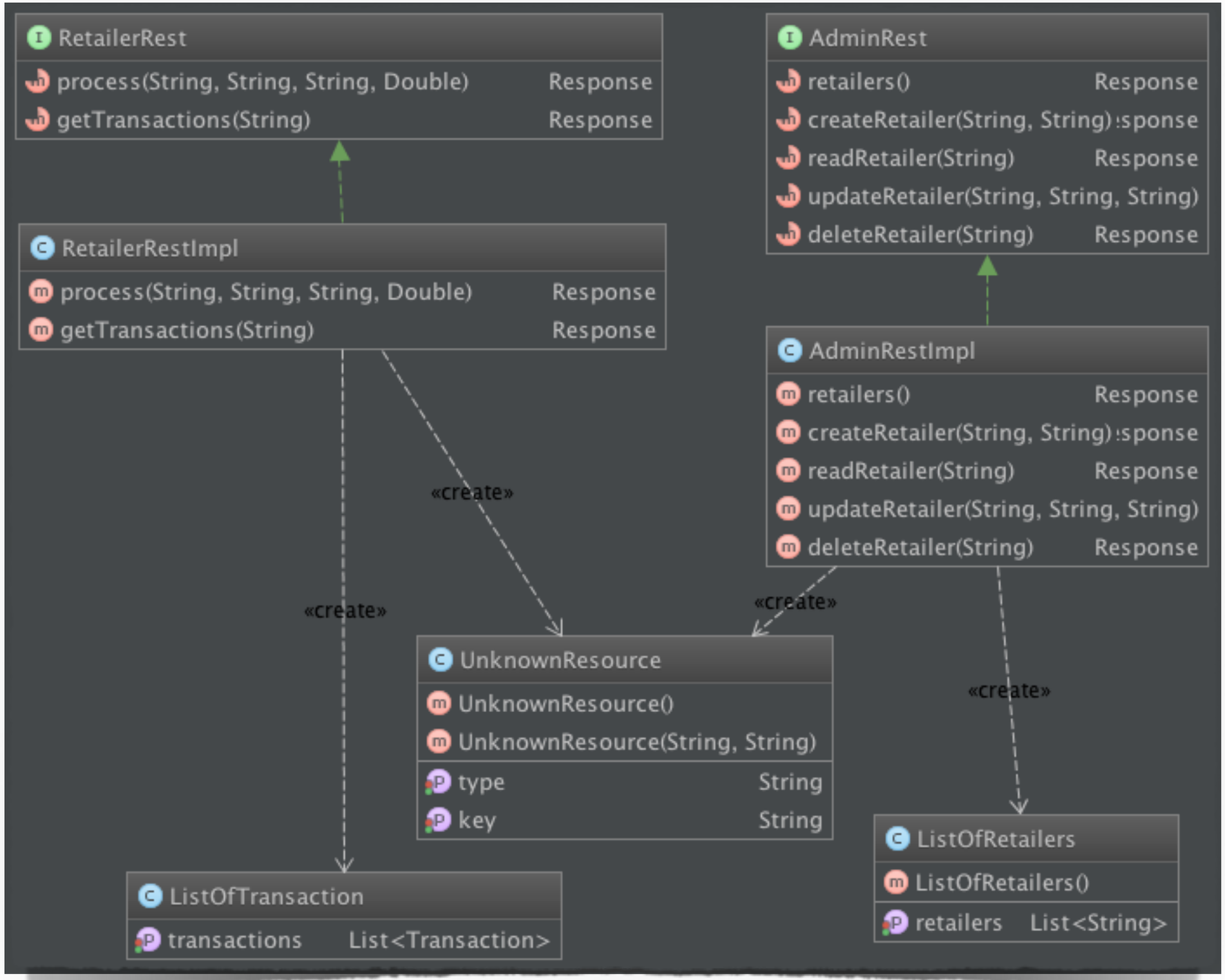
https://github.com/polytechnice-si/5A-2014-SOA1/tree/master/webservices/src/main/java/fr/unice/polytech/soa1/payment/doc

Resource Interface: Proposition

- **/rest/payment/public/{id}/transactions**:
  - **GET**: returns the transactions associated to the customer identified by the given id
- **/rest/payment/public/{id}/process**:
  - **POST**: the process used to perform a payment for customer id.

- **/rest/payment/private/retailers**:
  - **GET**: returns a list of links to registered retailers
  - **POST**: create a new retailer (status code: 201), available as a new resource
- **/rest/payment/private/retailers/{id}**:
  - **GET**: read the information stored for a given retailer
  - **PUT**: update an existing retailer with new information
  - **DEL**: delete an existing retailer

Resource Interface: Implementation



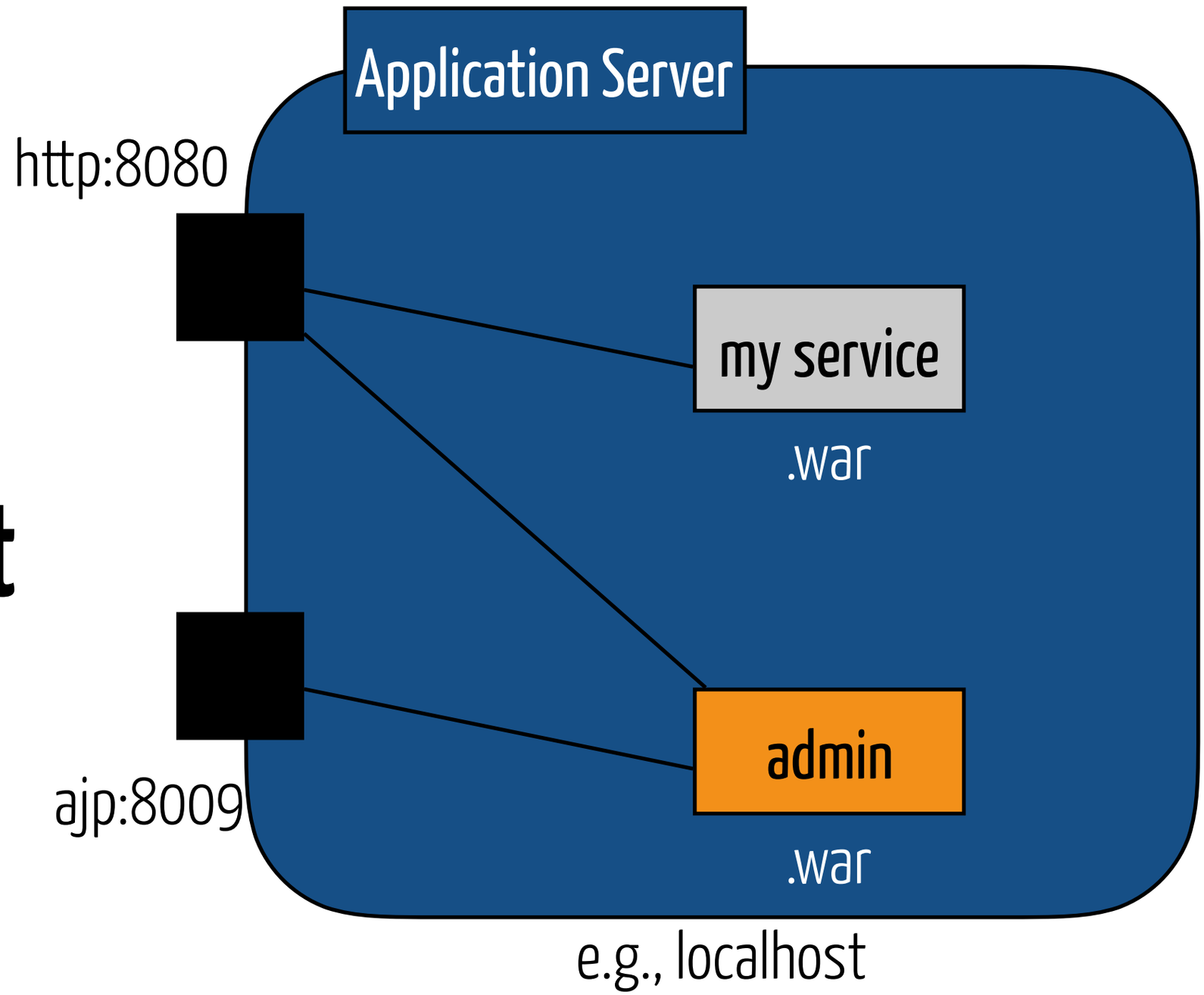
Bonus track

Services++

5

Hosting services

Service as deployable unit



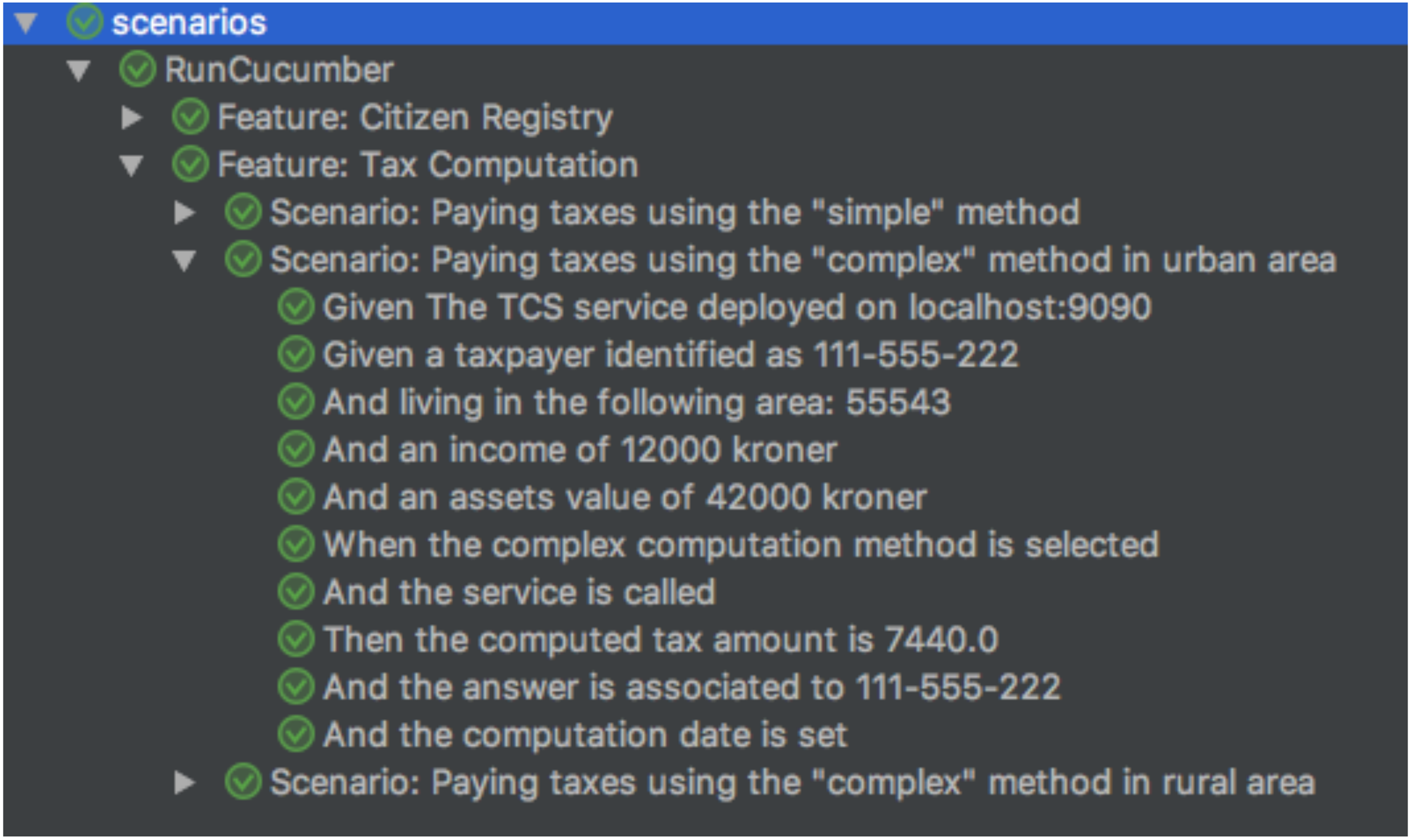
Acceptance Testing for Services

```
Scenario: Paying taxes using the "complex" method in urban area

  Given a taxpayer identified as 111-555-222
    And living in the following area: 55543
    And an income of 12000 kroner
    And an assets value of 42000 kroner
  When the complex computation method is selected
    And the service is called
 Then the computed tax amount is 7440.0
    And the answer is associated to 111-555-222
    And the computation date is set
```

Gherkin

```
@Given("^a taxpayer identified as (.*)$")
public void declare_a_taxpayer(String identifier) { this.taxPayerId = identifier; }


@Then("^the computed tax amount is (\\d+\\.\\d+)$")
public void validate_tax_amount(float value) {
    assertEquals(value, this.response.getAmount(),0.001);
}
```

JUnit

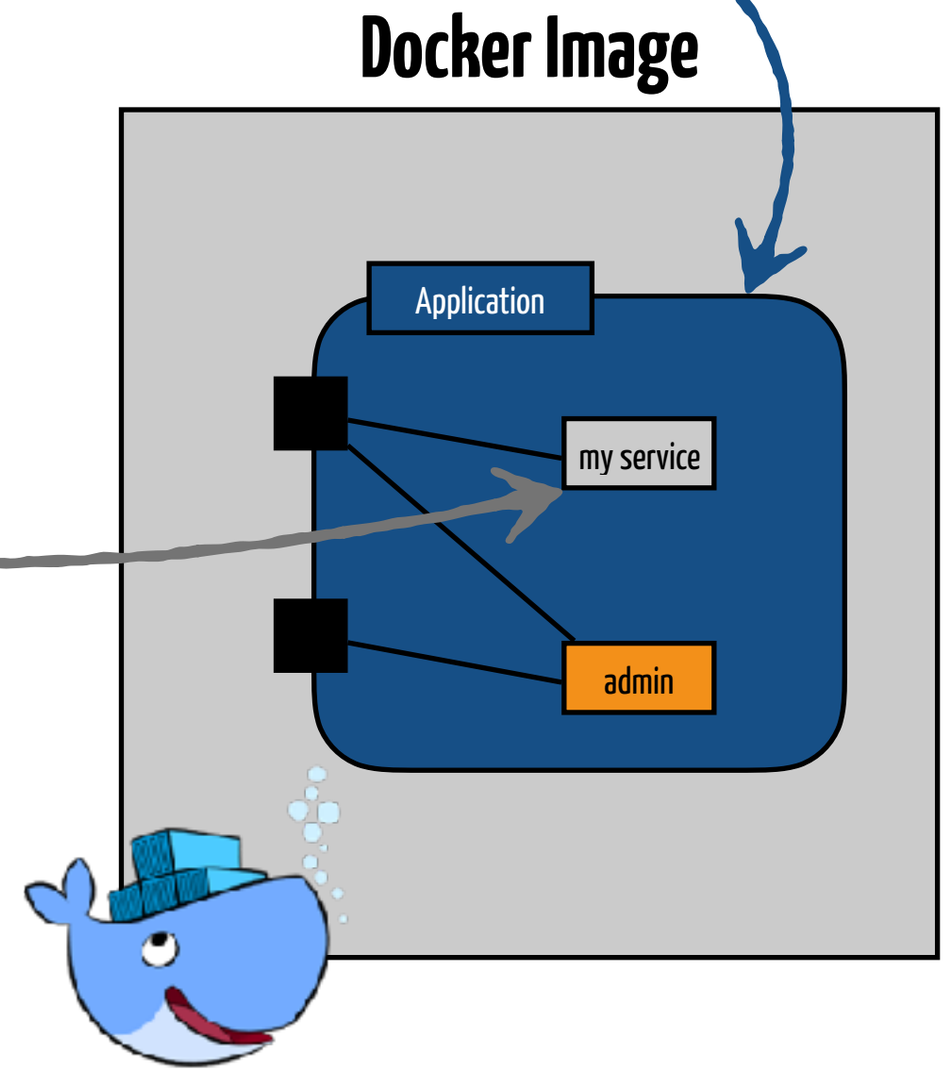


$ mvn verify

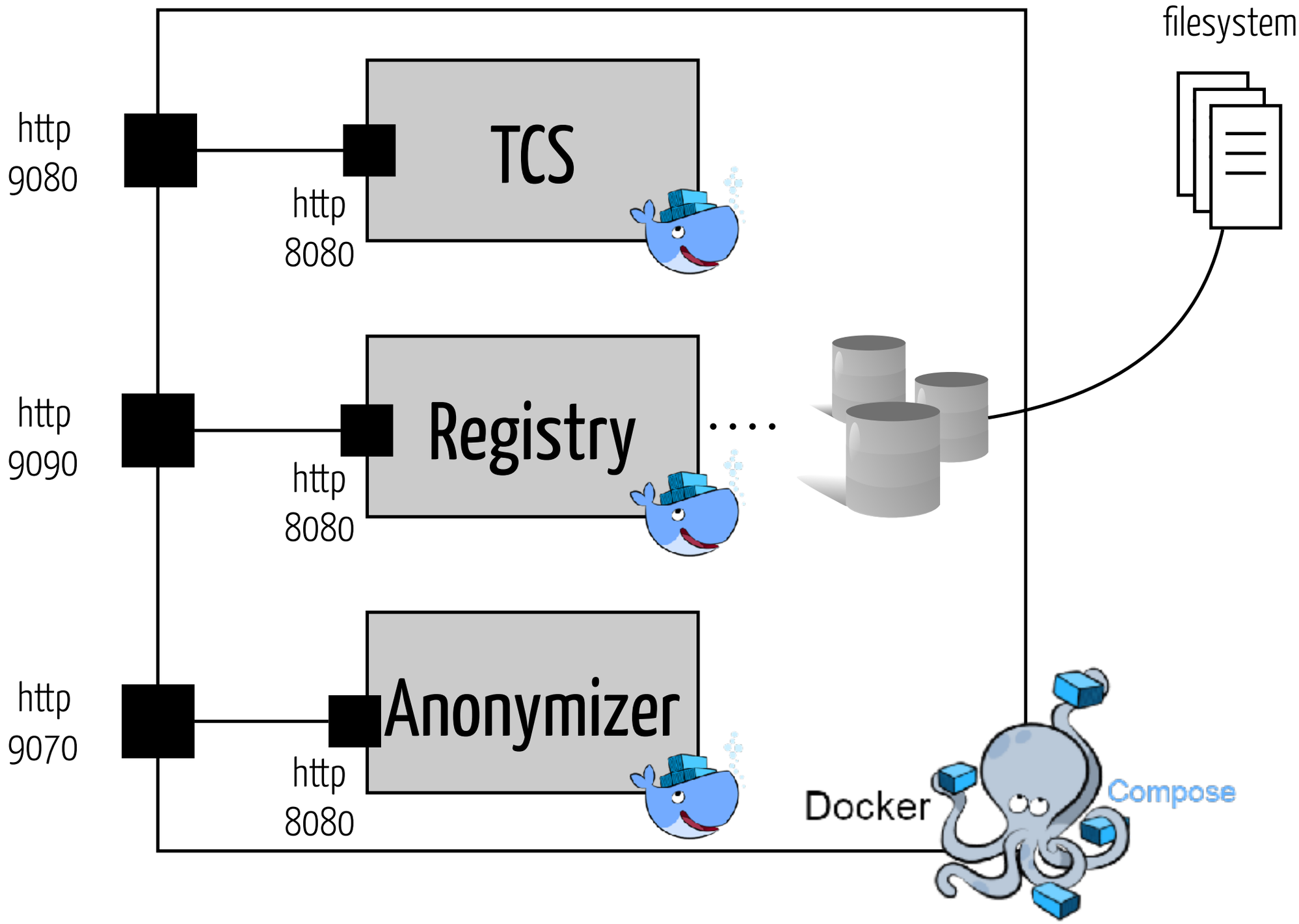
IDE Integration

Containerizing Services

```
FROM tomee:8-jre-7.0.3-plus
MAINTAINER Sébastien Mosser (mosser@i3s.unice.fr)

# Build with : docker build -t petitroll/tcs-rpc .
# Publish with: docker push petitroll/tcs-rpc

WORKDIR /usr/local/tomee/

COPY ./target/tcs-service-rpc.war ./webapps/.

EXPOSE 8080
```



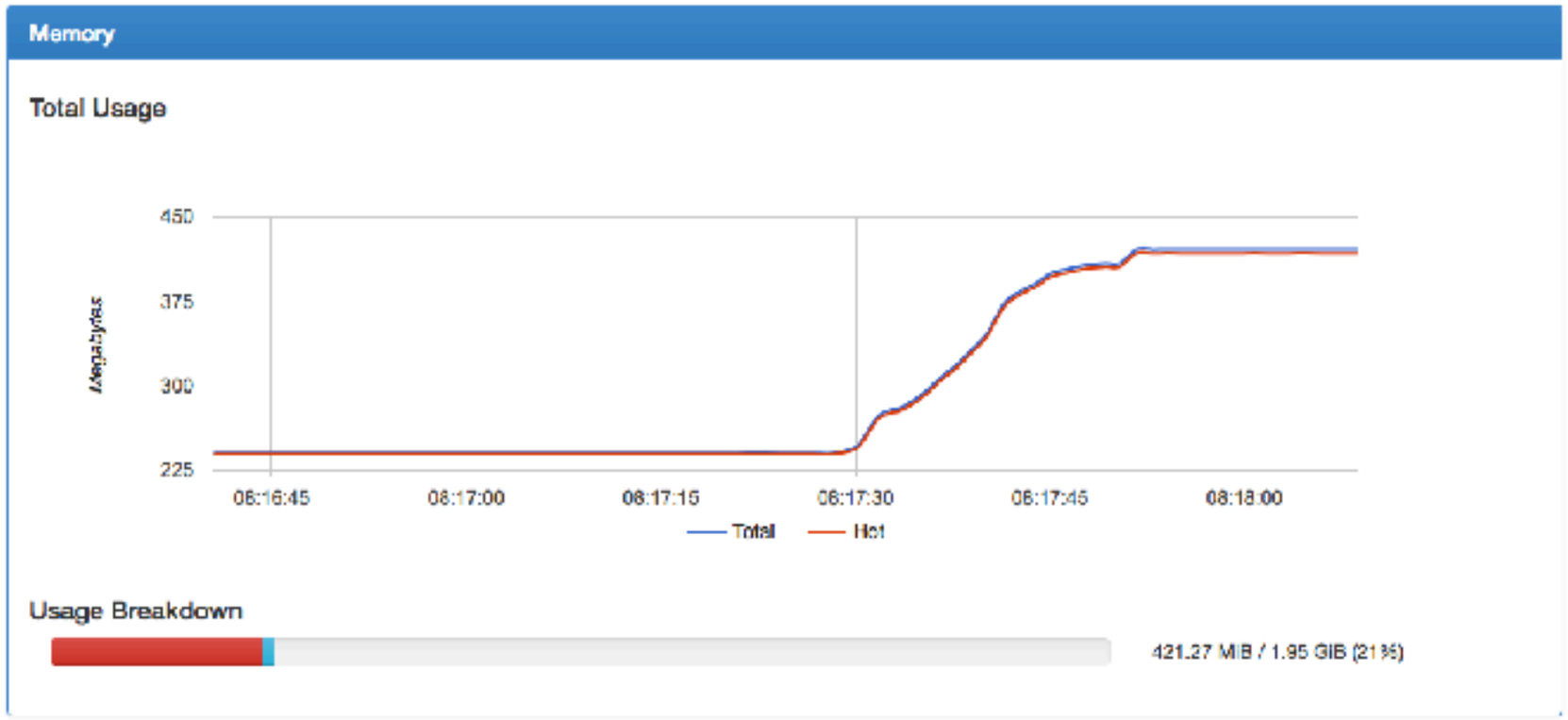
Composing Containers

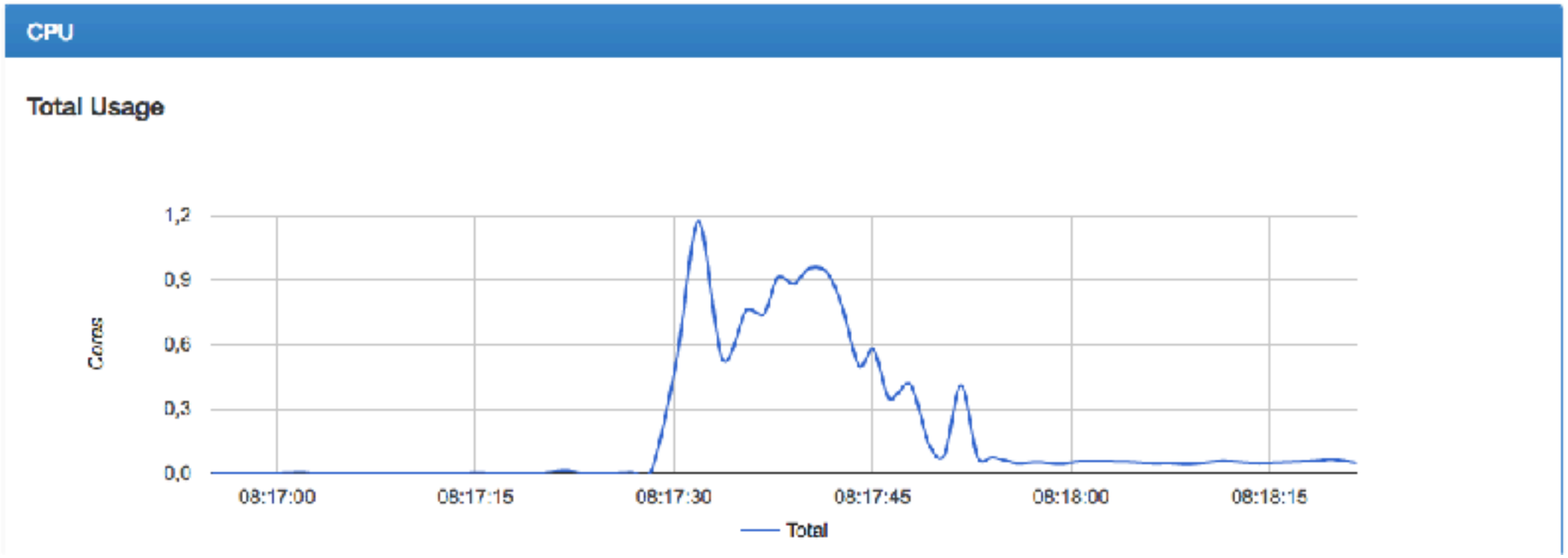


```yaml
tax-computation:                # Tax Computation, on port 9090
  container_name: tcs-computation
  image: petitroll/tcs-rpc
  ports:
    - "9090:8080"

id-generator:                   # ID generators, on port 9070
  container_name: tcs-generator
  image: petitroll/tcs-rest
  ports:
    - "9070:8080"

citizen-registry:               # Citizen registry, on port 9080
  container_name: tcs-citizens
  image: petitroll/tcs-doc
  depends_on:
    - database
  ports:
    - "9080:8080"

database:                       # MongoDB database, on port 27017
  container_name: tcs-database
  image: mongo:3.0
  volumes:
    - "./mongo_data:/data/db"
  ports:
    - "27017:27017"
```

$ docker-compose up -d

Monitoring Containers



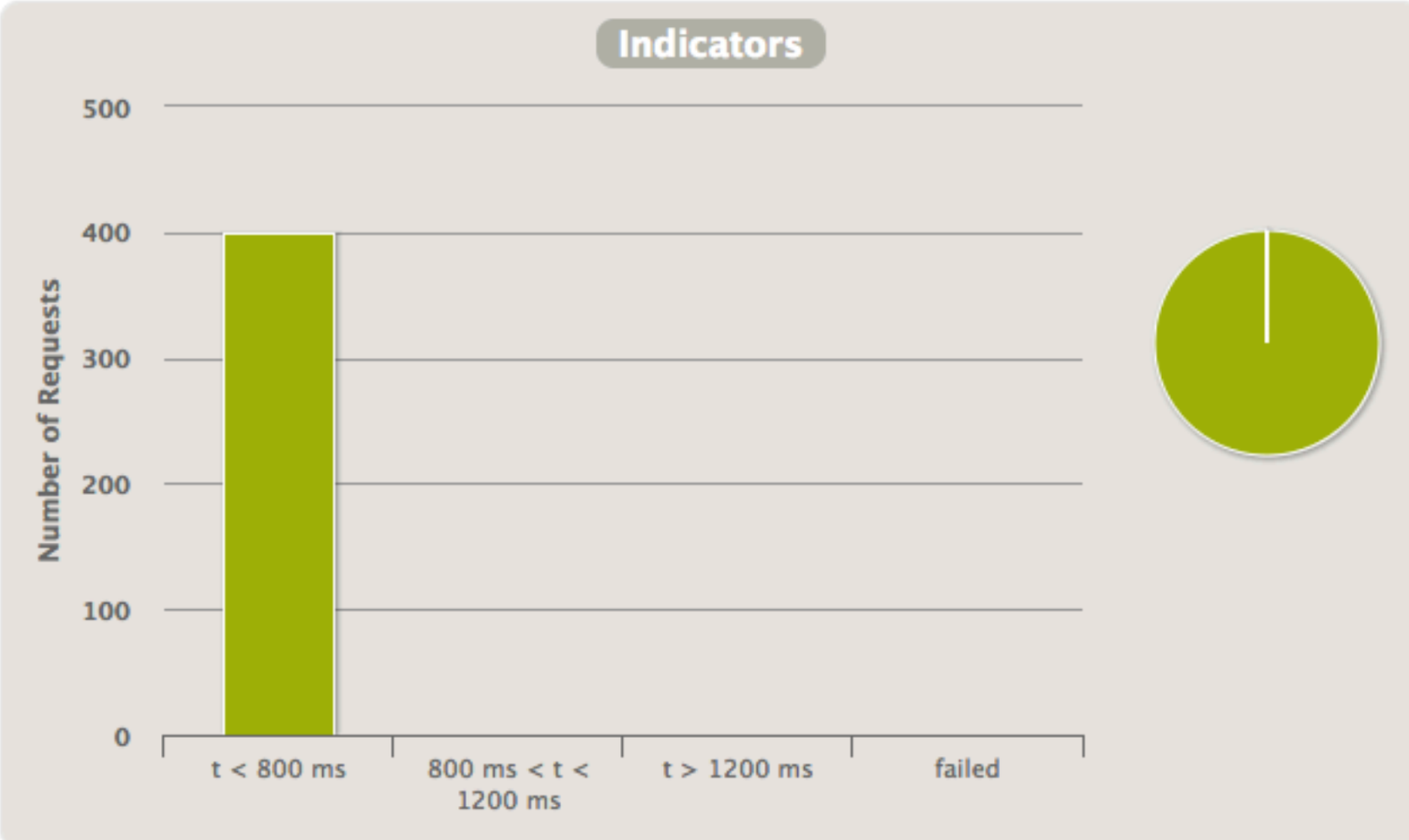


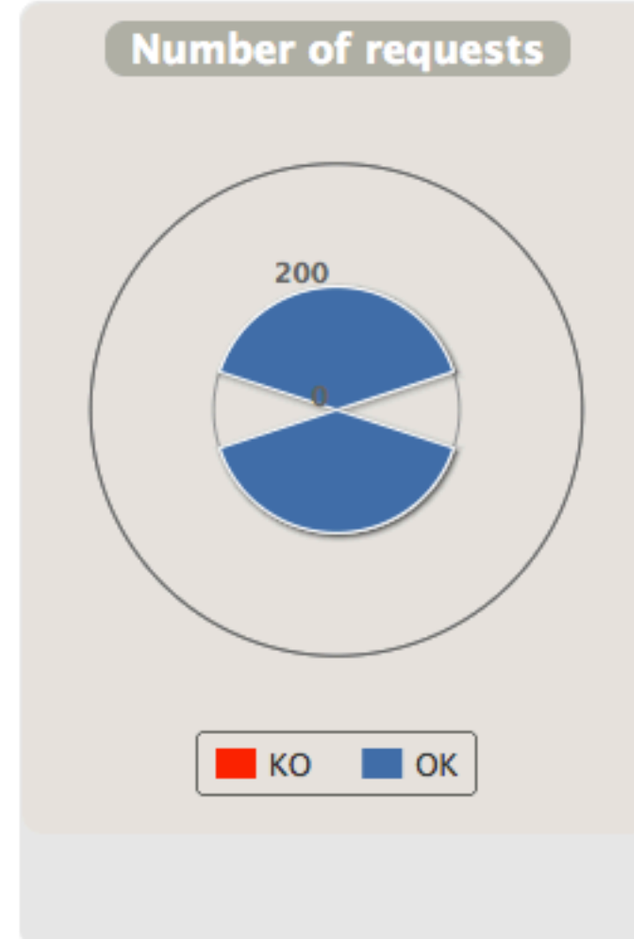
Stress Testing system

```
val stressSample =
  scenario("Registering Citizens")
      .repeat(10)
      {
        exec(session =>
          session.set("ssn", UUID.randomUUID().toString)
        )
          .exec(
            http("registering a citizen")
              .post("registry")
              .body(StringBody(session => buildRegister(session)))
              .check(status.is(200))
          )
          .pause(1 seconds)
          .exec(
            http("retrieving a citizen")
              .post("registry")
              .body(StringBody(session => buildRetrieve(session)))
              .check(status.is(200))
          )
      }
setUp(stressSample.inject(rampUsers(20) over (10 seconds)).protocols(httpConf))
```

$ mvn gatling:execute

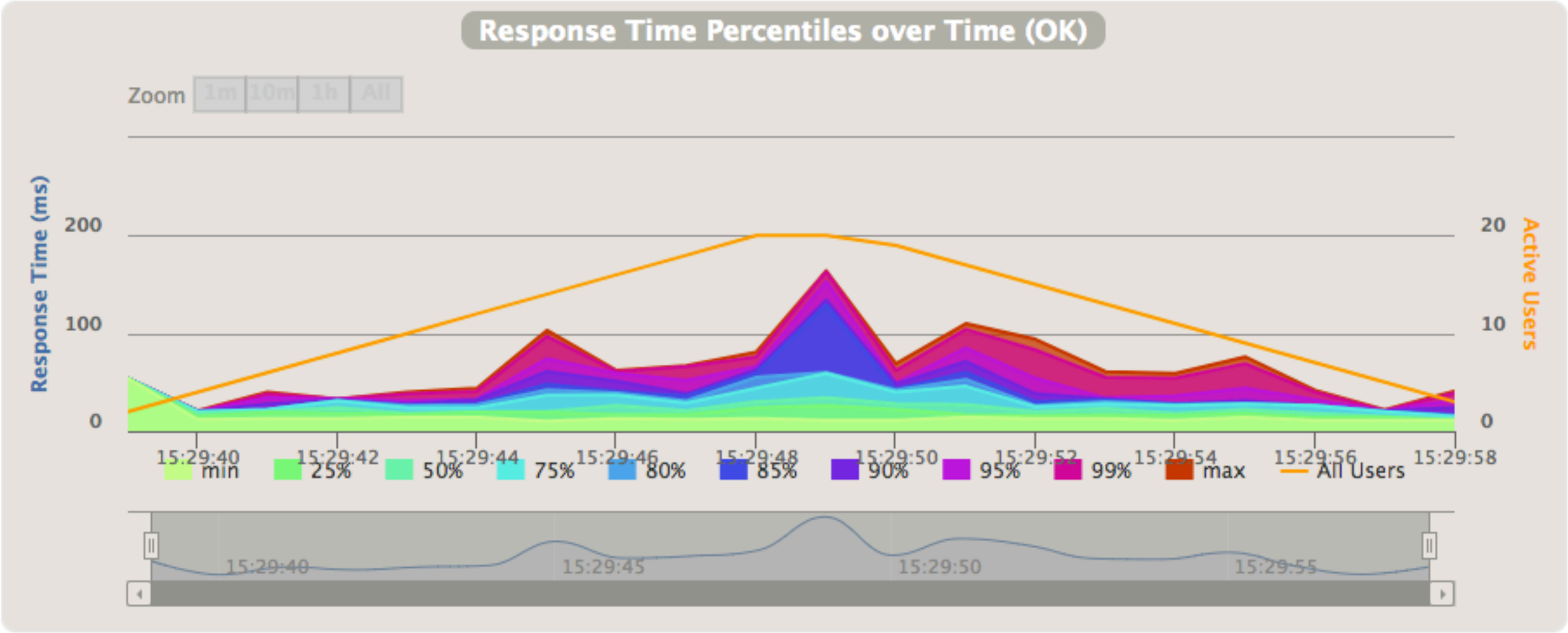
> Global Information

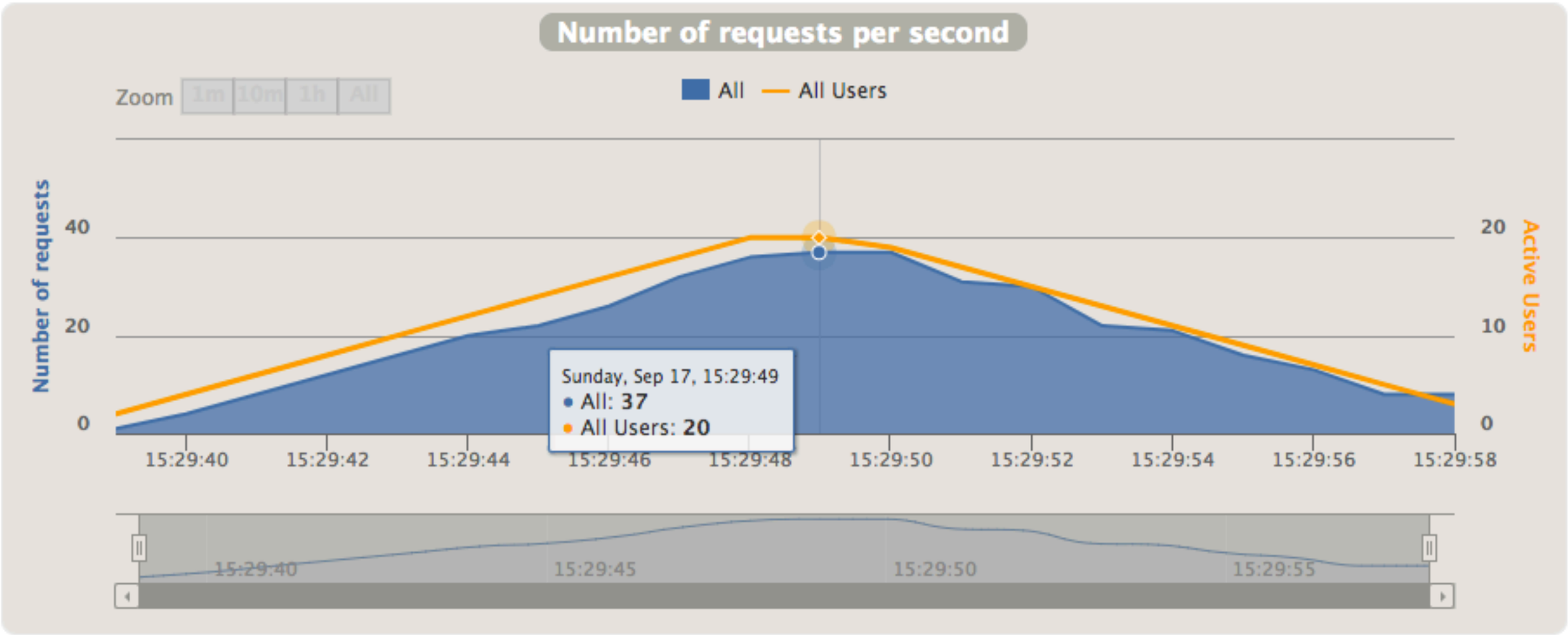




STATISTICS Expand all groups | Collapse all groups

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Req/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	400	400	0	0%	20	10	23	33	66	134	164	30	22
register... citizen	200	200	0	0%	10	11	24	36	67	152	164	32	24
retrievi... citizen	200	200	0	0%	10	10	22	32	61	103	135	28	18


Response Time Percentiles over Time (OK)
Zoom
1m
10m
1h
All
Response Time (ms)
200
100
0
20
10
0
Active Users
15:29:40
15:29:42
15:29:44
15:29:46
15:29:48
15:29:50
15:29:52
15:29:54
15:29:56
15:29:58
min
25%
50%
75%
80%
85%
90%
95%
99%
max
All Users
15:29:40
15:29:45
15:29:50
15:29:55


Number of requests per second
Zoom
1m
10m
1h
All
All
All Users
Number of requests
40
20
0
20
10
0
Active Users
Sunday, Sep 17, 15:29:49
All: 37
All Users: 20
15:29:40
15:29:42
15:29:44
15:29:46
15:29:48
15:29:50
15:29:52
15:29:54
15:29:56
15:29:58
15:29:40
15:29:45
15:29:50
15:29:55