Communication over the web for distributed systems with

REST APIs

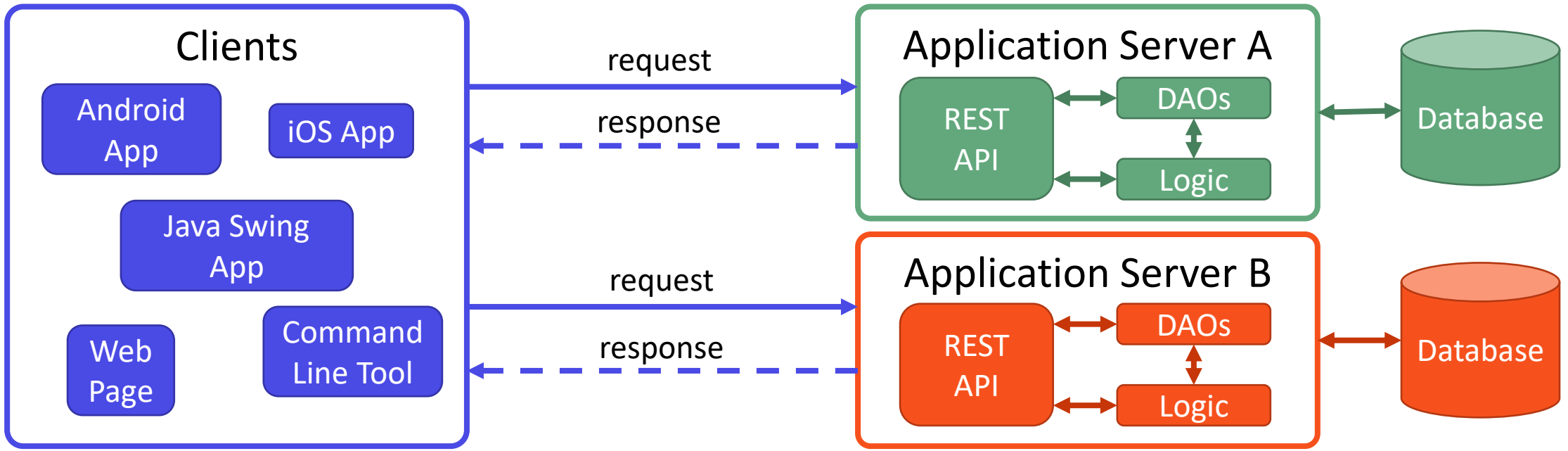
Luigi Libero Lucio Starace

https://luistar.github.io

luigiliberolucio.starace@unina.it

Architectural Background

- Clients and servers on different machines, communicate over WWW



Communication mechanisms

How can we handle communication between clients and applications?

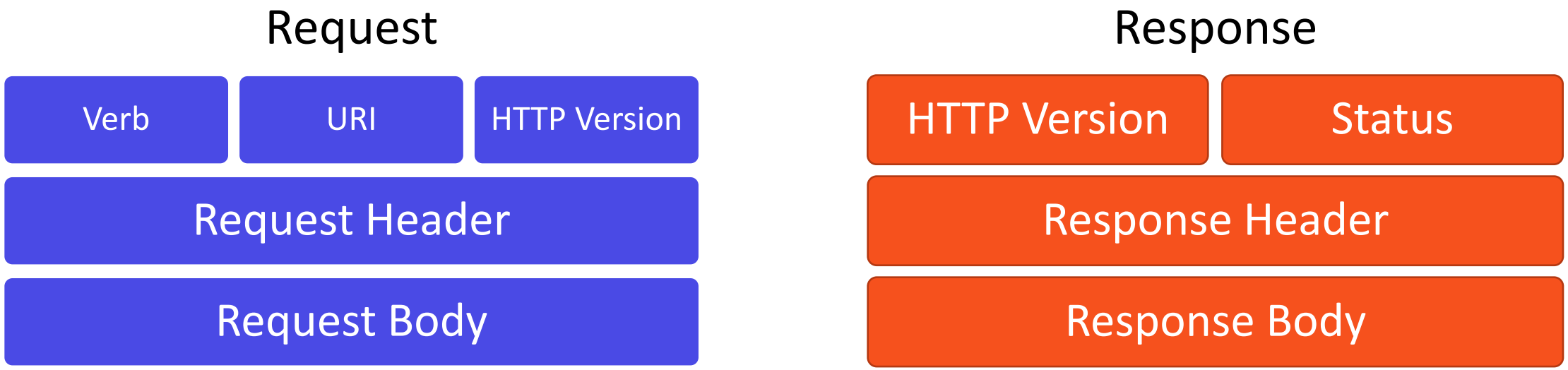
- Manually define a protocol over TCP, open sockets, etc…
  - Not very cost-effective, not very interoperable
- CORBA, Java RMI, SOAP, …
  - Dedicated protocols exist(ed), re-inventing an alternative to the web
- Just use web standards!

REST

- REST is a set of principles and guidelines that define how web standards should be used
- Based on HTTP and URIs
- Provides a common and consistent interface based on «proper» use of HTTP

HTTP

- HyperText Transfer Protocol
- Two types of messages: Request and Response



HTTP Requests

- HTTP can request different kinds of operations (using **Verbs**)
- The resource on which to operate is identified by the URI
- Based on idea of CRUD (Create, Retrieve, Update, Delete)

Verb	Operation Description
PUT	Here is some new data. Save it and **CREATE** a new resource
GET	**RETRIEVE** information about the resource
POST	Here is **UPDATE**d info about the resource
DELETE	**DELETE** this resource

HTTP Responses

- Status codes give information about the outcome of the request

Status Code	Meaning
200	Request was successfully handled
201	A resource was successfully created
400	Cannot understand the request
401	Authentication failed, or not authorized.
404	Resource not found
405	Method not supported by resource
500	Application error

HTTP: Data interexchange formats

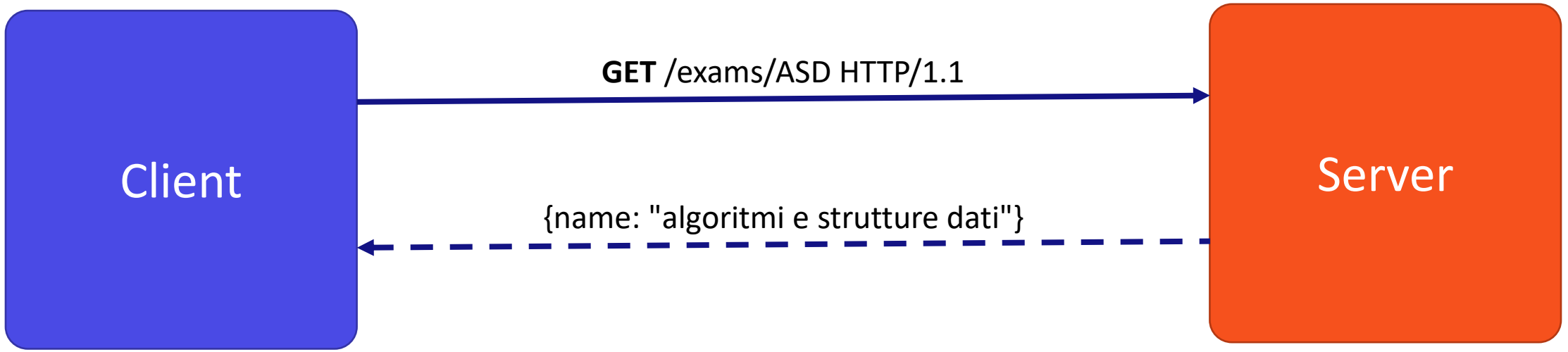
- Widely used formats include JSON and XML

```
{
   "exams": [{
         "name": "ASD",
         "grade": 30
      },{
         "name": "INGSW",
         "grade": 30
      }
   ]
}
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <exams>
    <exam>
      <name>ASD</name>
      <grade>30</grade>
    </exam>
    <exam>
      <name>INGSW</name>
      <grade>30</grade>
    </exam>
  </exams>
</root>
```

Representational State Transfer (REST)

- **Application State** (on the Client)
- **Resource State** (on the Server)
- Trasferred using appropriate representations (e.g.: JSON, XML, …)



REST Fundamentals

- A REST API allows to interact with **resources**
- All requests are associated to a unique URI
- «A resource is anything that's important enough to be referenced as a thing in itself.»[1]
- Resource typically (but not necessarily) correspond to persistent domain objects
- HTTP verbs should be used to retrieve or manipulate resources

[1] Richardson, Leonard, and Sam Ruby. *RESTful web services*. "O'Reilly Media, Inc.", 2008.

REST: Examples

We want to write an app that manages a list of Exams a student took

HTTP verb/URI	Meaning
GET /exams	Retrieve a list of all saved exams
GET /exams/<ID>	Retrieve only the exam whose ID is <ID>
POST /exam	Save a new Exam. Data of the exam to save are in the request body
PUT /exam/<ID>	Replace (or create) the exam whose ID is <ID> using the data in the request body
DELETE /exam/<ID>	Delete the exam having ID <ID>

A RESTful service with Spring Boot

- Practical Demo!

API Authentication/Authorization

- REST APIs allow users to manipulate resources
- In most cases, we don’t want that everyone is able to do so
- **Authentication**: We want that only legit users can access the resources
- **Authorization**: We may also want that some users can access only certain resources (e.g.: an employee shouldn’t be able to update its own salary)

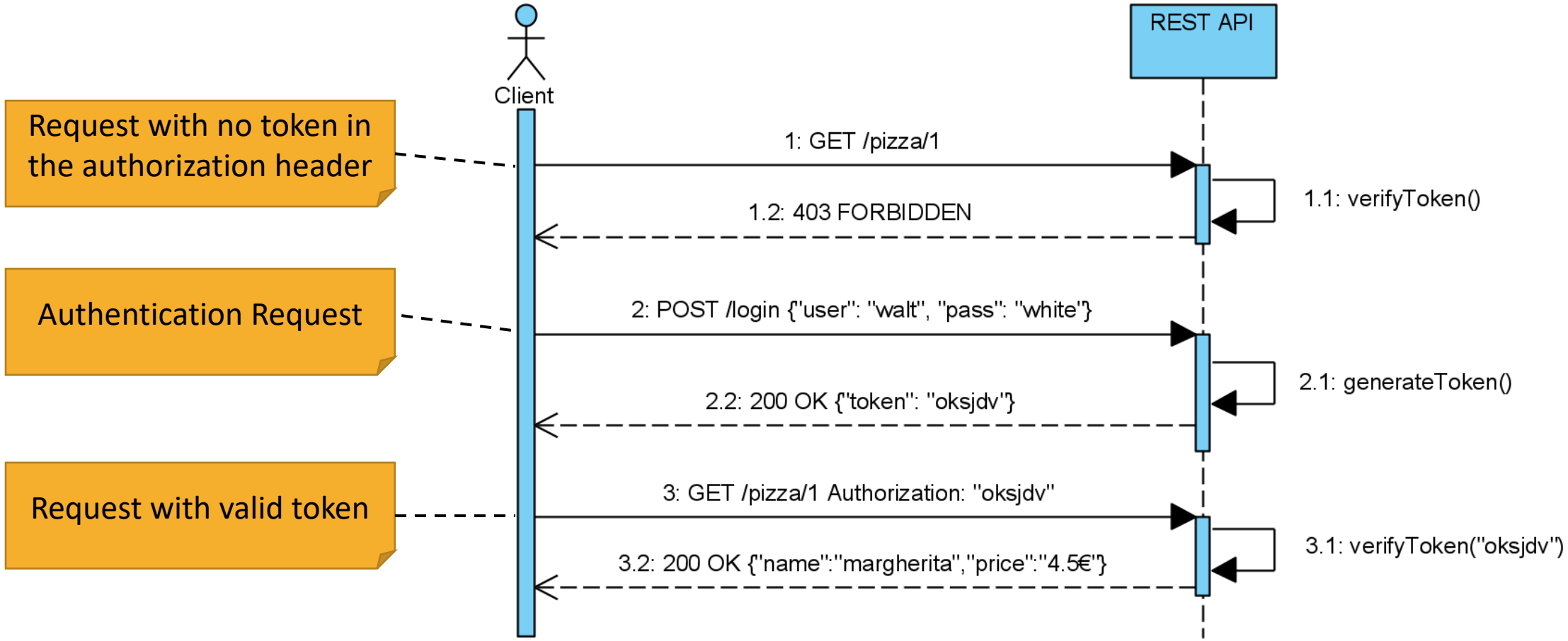
How to secure REST APIs

A widely-used authentication scheme is based on Tokens

💡 Idea:

1. Clients send a request with username and password to the API
2. API validates username and password, and generates a token
3. The token (a string) is returned to the client
4. Client must pass the token back to the API at every subsequent request that requires authentication (in the Authorization Header)
5. API verifies the token before responding

Token-based Authentication Scheme



JSON Web Token (JWT)

- JWT is a widely-adopted open standard (RFC 7519)
- Allows to securely share claims between two parties
- A JWT token is a string consisting of three parts, separated by “.”
- Structure: Header.Payload.Signature

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX
VCJ9.eyJuYW1lIjoibHVpZ2kiLCJyb2x
lIjoiYWRtaW4iLCJleHAiOjE2NzAzOTg
0MzJ9.fopBYrax8wcB7rnPjCcOMc62IT
2lJdvyOdyixMWMZAQ
```

JSON Web Token