

RESTful Services

Didier DONSEZ
Université Grenoble Alpes
Polytech Grenoble & LIG ERODS

Outline

Principles

Richardson Maturity Model

Design Patterns

Data Formats

Documentation, Validation

Frameworks

Swagger, JAX-RS (JavaEE), Springfox ...

Security

OAuth2.0

REST = REpresentational State Transfert

- Style d'architecture de type client-serveur
 - 3 Levels of Richardson Maturity Model
 - Best practices http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf
- Considère des collections de ressources (maintenant un état interne)
 - Adressables pour une URI
 - `http://www.boeing.com/aircraft` `http://www.boeing.com/aircraft/747`
 - Accessible via les méthodes (verb) HTTP (CRUD)
 - PUT, GET, POST, DELETE, HEAD, OPTION, PATCH
 - Status code (200, 201, 409, ...), Header (Accept, Location, ...)
 - Représentation formatée en fonction du client
 - Formats : XML, HTML (fragment), JSON, JPEG,...
- Outils
 - Caches, Proxies, Load Balancers ...

Web Routing

«Classic » (Level 0) → Réponse : text/html

GET http://example.com/book?action=showall

GET http://example.com/book?action=show&isbn=0596529260

GET http://example.com/book?action=modify&isbn=0596529260&price=49.99

POST http://example.com/book?action=modify

GET http://example.com/book?action=delete&isbn=0596529260

REST (Level 2) → Response : application/json, text/xml, text/csv

GET http://example.com/book

GET http://example.com/book/0596529260

POST http://example.com/book/0596529260

PUT http://example.com/book/0596519260

DELETE http://example.com/book/0596519260

Exercice : How to implement a (stateful) customer shopping cart ?

One solution

POST http://example.com/cart

PUT http://example.com/cart/676564542453452455435624534562

Richardson Maturity Model

- Level 0
 - Communication client / serveur via le protocole HTTP
 - 1 URL / 1 type de verbe HTTP (en général POST)
- Level 1: Resources
 - Plusieurs URLs hiérarchisées
 - Représente des collections
 - toujours 1 seul verbe HTTP (en général POST)
- Level 2: HTTP Verbs
 - Verbes HTTP (GET, POST, PUT, DELETE, OPTION, HEAD, PATCH)
 - Status code HTTP
- Level 3: Hypermedia Controls
 - HATEOAS (Hypertext As The Engine Of Application State)
 - URL de navigation dans les ressources
- A lire <http://martinfowler.com/articles/richardsonMaturityModel.html>

CRUD et Verbes REST (RMM Level 2)

- Create
 - PUT if and only if you are sending the full content of the specified resource (URL).
 - POST if you are sending a command to the server to create a subordinate of the specified resource, using some server-side algorithm.
- Retrieve = GET, HEAD.
- Update
 - PUT if and only if you are updating the full content of the specified resource.
 - POST if you are requesting the server to update one or more subordinates of the specified resource.
 - PATCH for semantic changes
- Delete = DELETE.
- Info = OPTION
 - Used to request information about the communication options of the resource you are interested in. It allows the client to determine the capabilities of a server and a resource without triggering any resource action or retrieval.

Status Code des réponses REST

- 200 OK - Response to a successful GET, PUT, PATCH or DELETE. Can also be used for a POST that doesn't result in a creation.
 - 201 Created - Response to a POST that results in a creation. Should be combined with a Location header pointing to the location of the new resource
 - 204 No Content - Response to a successful request that won't be returning a body (like a DELETE request)
 - 304 Not Modified - Used when HTTP caching headers are in play
 - 400 Bad Request - The request is malformed, such as if the body does not parse
 - 401 Unauthorized - When no or invalid authentication details are provided. Also useful to trigger an auth popup if the API is used from a browser
 - 403 Forbidden - When authentication succeeded but authenticated user doesn't have access to the resource
 - 404 Not Found - When a non-existent resource is requested
 - 405 Method Not Allowed - When an HTTP method is being requested that isn't allowed for the authenticated user
 - 410 Gone - Indicates that the resource at this end point is no longer available. Useful as a blanket response for old API versions
 - 415 Unsupported Media Type - If incorrect content type was provided as part of the request
 - 422 Unprocessable Entity - Used for validation errors
 - 429 Too Many Requests - When a request is rejected due to rate limiting
- From <http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api>

HATEOAS

Hypertext As The Engine Of Application State

API de navigation dans les collections et les ressources

- Martin Fowley, *RMM Level 3 introduces discoverability, providing a way of making a protocol more self-documenting.*

•

GET /account/12345 HTTP/1.1

HTTP/1.1 200 OK

```
<?xml version="1.0"?>
```

```
<account>
```

```
<account_number>12345</account_number>
```

```
<balance currency="usd">100.00</balance>
```

```
<link rel="self" href="/account/12345" />
```

```
<link rel="deposit" href="/account/12345/deposit" />
```

```
<link rel="withdraw" href="/account/12345/withdraw" />
```

```
<link rel="transfer" href="/account/12345/transfer" />
```

```
<link rel="close" href="/account/12345/close" />
```

```
<link rel="comment" type="application/x.atom+xml" title="Blog comments" href="/account/12345/comment"/>
```

```
</account>
```

REST Asynchronous Requests (i)¶

- Opérations différées ou de longue durée (*long-live*) sur des ressources/collections REST

POST */blog* HTTP/1.1

```
<blog><title>Bla Bla</title><author>Me</author><text>Bla Bla Bla Bla !  
</text></blog>
```

HTTP/1.1 202 Accepted

Location: </queue/12345>

GET </queue/12345> HTTP/1.1

HTTP/1.1 200 OK

```
<queue>
```

```
<status>Pending</status>
```

```
<eta>10 minutes</eta>
```

```
<link rel="cancel" method="delete" href="/queue/12345" />
```

```
</queue>
```

...

verbe DELETE
pour annuler l'opération

- <http://restcookbook.com/Resources/asynchroneous-operations/>
- <https://www.adayinthelifeof.nl/2011/06/02/asynchronous-operations-in-rest/>

REST Asynchronous Requests (ii)¶

...

```
GET /queue/12345 HTTP/1.1
```

```
HTTP/1.1 200 OK
```

```
<queue>
```

```
<status>In progress</status>
```

```
<eta>3 minutes, 25 seconds</eta>
```

```
</queue>
```

Plus de verbe DELETE
pour annuler l'opération

```
GET /queue/12345 HTTP/1.1
```

```
HTTP/1.1 303 See Other
```

```
Location: /blog/20140101-Bla%20Bla
```

- <http://restcookbook.com/Resources/asynchroneous-operations/>
- <https://www.adayinthelifeof.nl/2011/06/02/asynchronous-operations-in-rest/>

REST PATCH verb

PUT versus PATCH

PUT fournit la nouvelle valeur intégrale de l'entité

PATCH donne des instructions à appliquer

JSON Patch <http://tools.ietf.org/html/rfc6902>

```
PATCH /users/123
```

```
Content-Type: application/json-patch
```

```
[  
  { "op": "replace", "path": "/email", "value": "new.email@example.org" }  
]
```

```
PATCH /agenda/123
```

```
Content-Type: application/json-patch+json
```

```
If-Match: "abc123"
```

```
[  
  { "op": "test", "path": "/a/b/c", "value": "foo" },  
  { "op": "remove", "path": "/a/b/c" },  
  { "op": "add", "path": "/a/b/c", "value": [ "foo", "bar" ] },  
  { "op": "replace", "path": "/a/b/c", "value": 42 },  
  { "op": "move", "from": "/a/b/c", "path": "/a/b/d" },  
  { "op": "copy", "from": "/a/b/d", "path": "/a/b/e" }  
]
```

Si /a/b/c!= foo,
les opérations suivantes
ne sont pas réalisées

207 Multi-Status

<http://williamdurand.fr/2014/02/14/please-do-not-patch-like-an-idiot/>

REST

Safety and Idempotence

- Idempotency is important in building a fault-tolerant API

Verb	Idempotent	Safe
OPTION	yes	yes
GET	yes	yes
HEAD	yes	yes
PUT	yes	no
POST	no	no
DELETE	yes	no
PATCH	no	no

REST

Paginate collection

- Motivation
 - Limit transfer of large set of subresources
-
- GET /collection HTTP/1.1
- HTTP/1.1 200 OK
 - <collection>
 - <article id="1">...</article> <article id="2">...</article> <article id="3">...</article> <article id="4">...</article> <article id="5">...</article>
 - <meta>
 - <link rel="self" href="/collection" title="Current page"/> <link rel="next" href="/collection/2" title="Next page"/> <link rel="first" href="/collection/1" title="First page"/> <link rel="last" href="/collection/5" title="Last page"/>
 - </meta>
 - </collection>
- See more at: <http://restcookbook.com/Resources/pagination/#sthash.PSefZe0Q.dpuf>

REST

Paginate collection

- Range

```
OPTIONS /api/collection HTTP/1.1
```

```
HTTP/1.1 200 OK
```

```
Accept-Ranges: resources
```

```
GET /api/collection
```

```
Range: resources=100-199
```

- REST Worst Practices

<http://jacobian.org/writing/rest-worst-practices/>

REST Caching

- 2 approaches
- Etag
 - When generating a request, include a HTTP header ETag containing a hash or checksum of the representation. This value should change whenever the output representation changes. Now, if an inbound HTTP requests contains a If-None-Match header with a matching ETag value, the API should return a 304 Not Modified status code instead of the output representation of the resource.
- Last-Modified
 - This basically works like to ETag, except that it uses timestamps. The response header Last-Modified contains a timestamp in RFC 1123 format which is validated against If-Modified-Since. Note that the HTTP spec has had 3 different acceptable date formats and the server should be prepared to accept any one of them.

REST Notification

- Notification (long poll)
 - Example : return at least one resource with an ID > 100.
GET /api/collection
Range: 100-
Expect: nonempty-response
- Status code
 - 416 REQUEST RANGE NOT SATISFIABLE
 - 413 Requested Entity Too Large
 - 206 PARTIAL CONTENT
- REST Worst Practices <http://jacobian.org/writing/rest-worst-practices/>

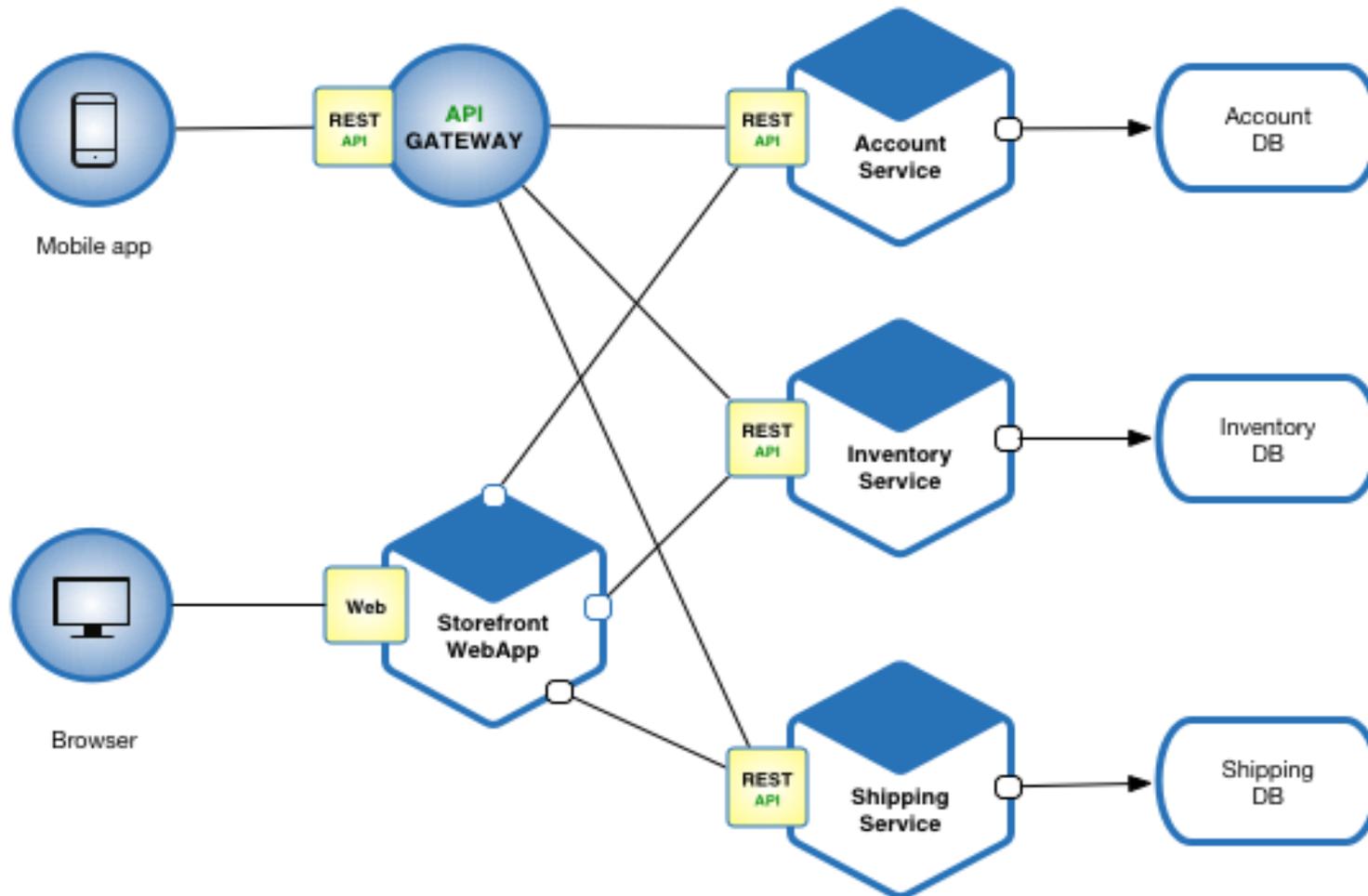
REST API and Cookies

- Not a best practice for passing parameter
 - But existing cookie-based (legacy) APIs
 - Remark : OpenAPI
 - The location of the parameter can be "query", "header", "path" or "cookie".

API Throttling and Rate Limiting

- Motivation
 - Limit the number of requests per user profiles
 - Avoid server collapsed and spikes
 - DDoS
- HTTP Responses
 - Status Code : 429 Too Many Requests
 - Headers : X-Rate-Limit-Limit, X-Rate-Limit-Remaining, X-Rate-Limit-Reset
- Algorithms
 - Token Bucket ...
- Frameworks
 - RateLimitJ, Spring Rate Limit, Resilient4J, ...
 - Distributed : using on Redis, Consul, Zookeeper, Hazelcast, ...

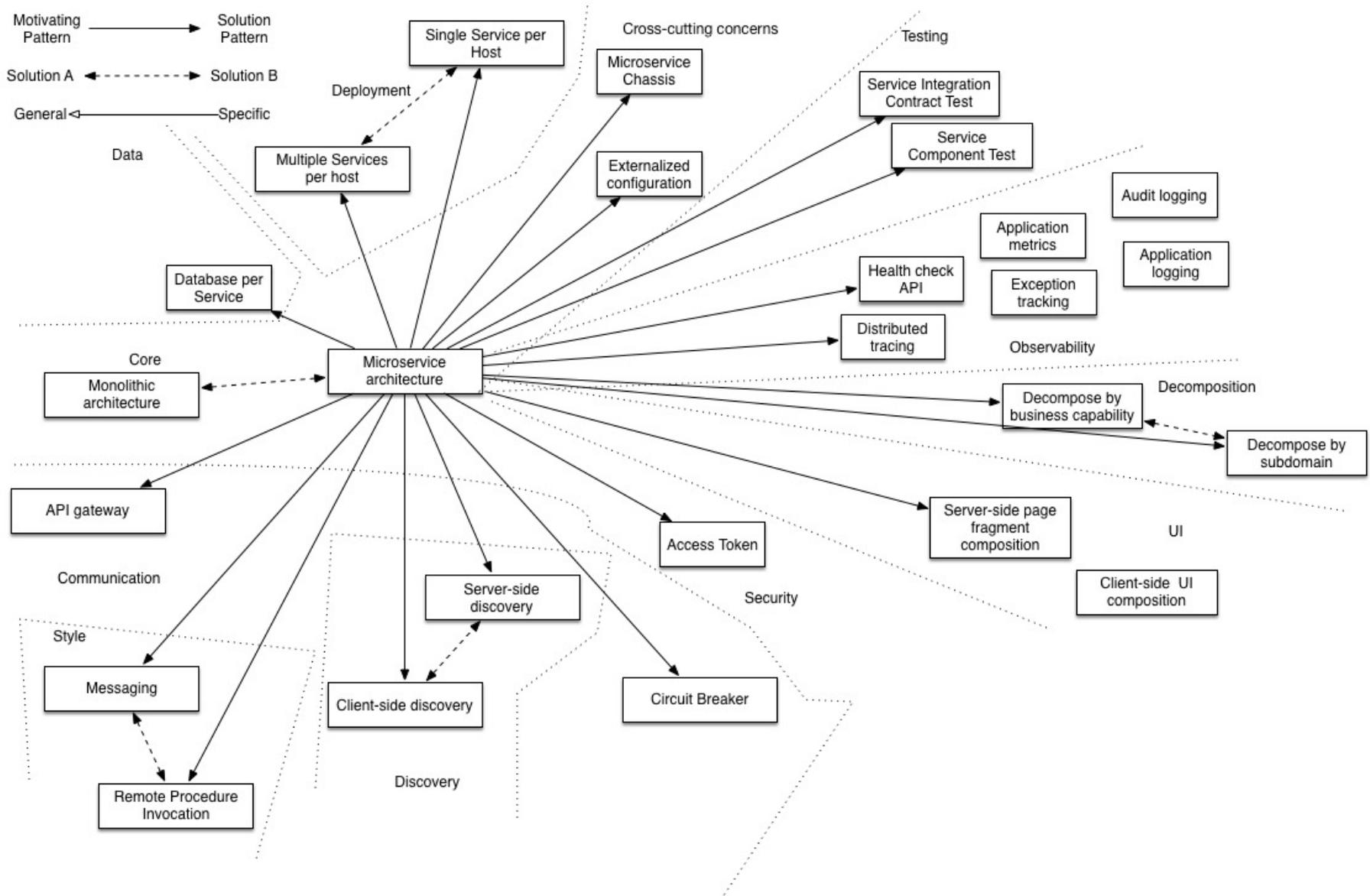
Microservice Architecture



Saga of Microservices

- Problem : data consistency of several microservices

Microservices ecosystem



Netflix Zuul

- Voir VT2017

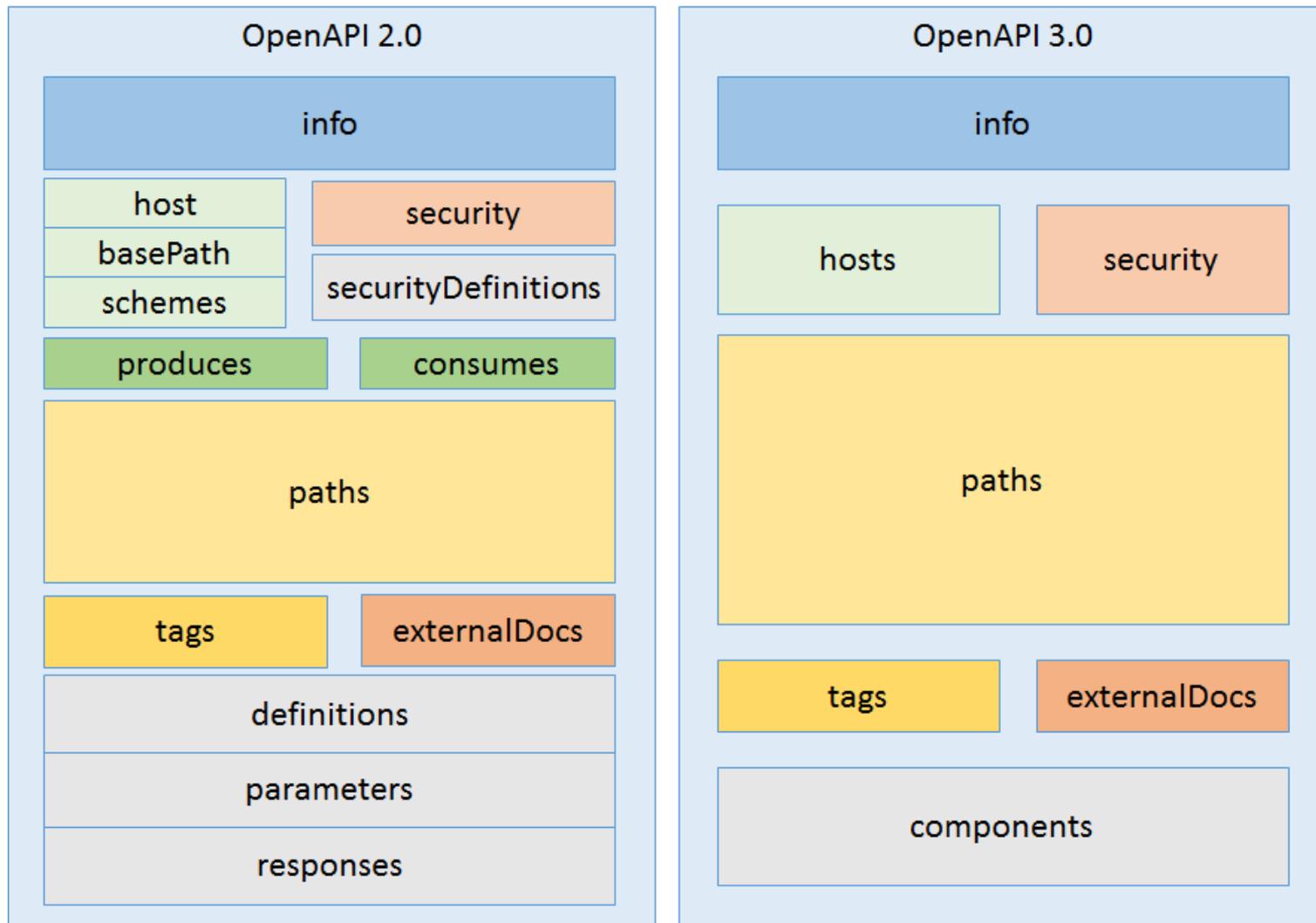
Modélisation / Documentation de services RESTFul

- OpenAPI (Swagger)
 - YAML, JSON
 - Documentation, Test, Editeur live, Génération multi-langages
 - Utilisé par des nombreuses entreprises
 - Supporté par la LinuxFoundation
- RESTful API Modeling Language (RAML)
 - YAML
 - <https://github.com/raml-org/raml-spec/blob/master/versions/raml-10/raml-10.md/>
- Enuciate

OAI OpenAPI OPEN API INITIATIVE

- Describes RESTFul API
- Linux Foundation

OAI OpenAPI v3



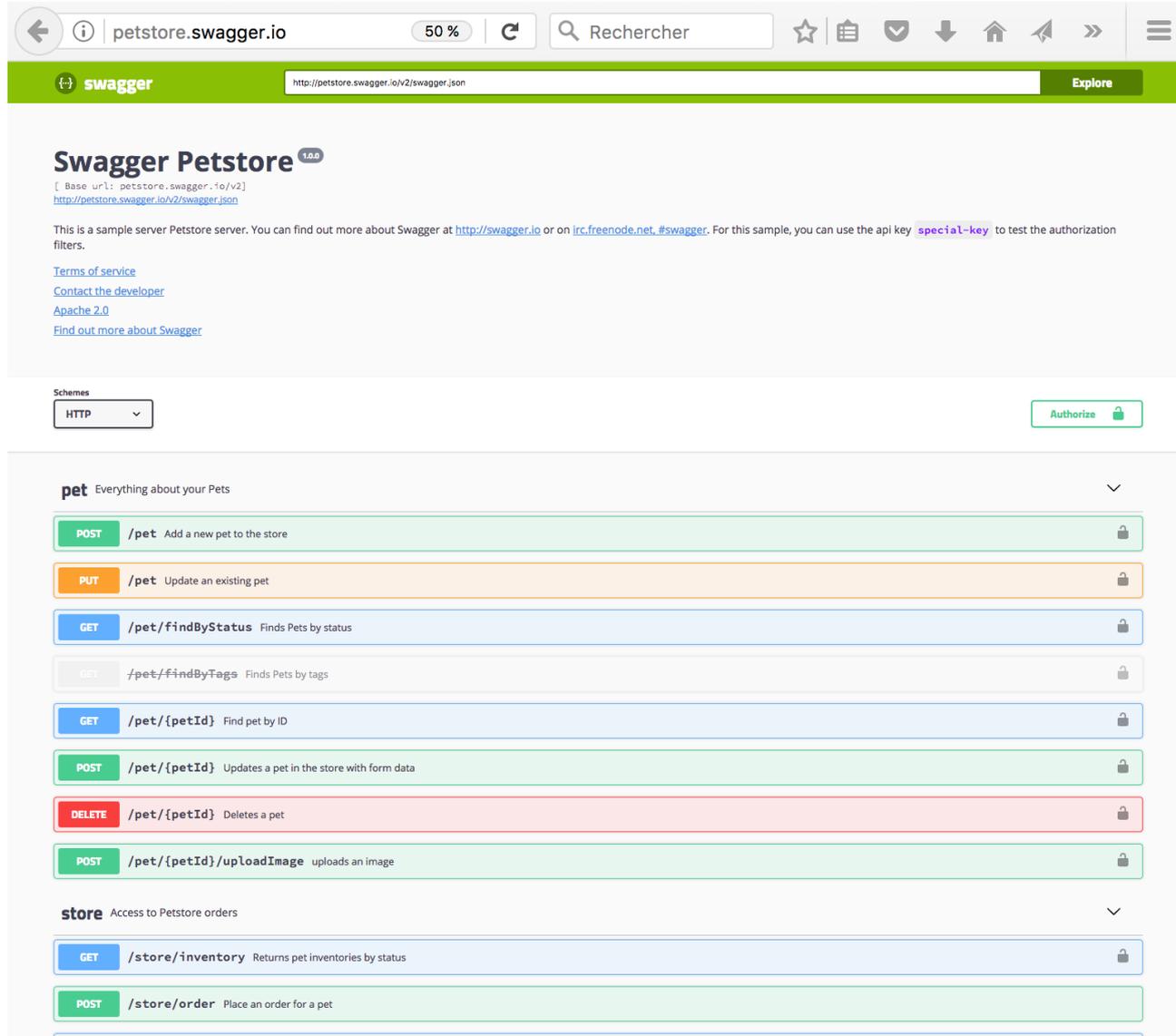
OAI OpenAPI v3

-
- Part 1: Background and how to get involved.
- Part 2: Structural Changes
- Part 3: Request Parameters
- Part 4: Protocol and Payload
- Part 5: Documentation
-

API and WebHooks

Swagger UI

- Documentation et Invocation/Test



The screenshot shows the Swagger UI interface for the Swagger Petstore API. The browser address bar displays `petstore.swagger.io` at 50% zoom. The Swagger logo and the URL `http://petstore.swagger.io/v2/swagger.json` are visible in the top navigation bar, along with an **Explore** button.

The main content area features the title **Swagger Petstore** with a **1.0.0** version tag. Below the title, the base URL is specified as `[Base url: petstore.swagger.io/v2]` with a link to the JSON file. A descriptive paragraph explains that this is a sample server and provides instructions on how to use the `special-key` API key for authorization. Links for [Terms of service](#), [Contact the developer](#), [Apache 2.0](#), and [Find out more about Swagger](#) are provided.

Below the text, there is a **Schemes** dropdown menu set to **HTTP** and an **Authorize** button with a lock icon.

The API endpoints are organized into two sections:

- pet** Everything about your Pets (indicated by a dropdown arrow):
 - POST** `/pet` Add a new pet to the store
 - PUT** `/pet` Update an existing pet
 - GET** `/pet/findByStatus` Finds Pets by status
 - GET** `/pet/findByTags` Finds Pets by tags
 - GET** `/pet/{petId}` Find pet by ID
 - POST** `/pet/{petId}` Updates a pet in the store with form data
 - DELETE** `/pet/{petId}` Deletes a pet
 - POST** `/pet/{petId}/uploadImage` uploads an image
- store** Access to Petstore orders (indicated by a dropdown arrow):
 - GET** `/store/inventory` Returns pet inventories by status
 - POST** `/store/order` Place an order for a pet

Swagger Editor

Edition + Test + Génération de test

The image shows the Swagger Editor interface. On the left, a code editor displays a Swagger specification for a PetStore API. The specification includes version information, a title, a description, host, base path, schemes, consumes, produces, and several endpoints. The right side of the interface shows the rendered API documentation, including the title 'PetStore on Heroku', base URL, and a list of endpoints with their methods and paths.

```
1 swagger: '2.0'
2 info:
3   version: 1.0.0
4   title: PetStore on Heroku
5   description: |
6     **This example has a working backend hosted in Heroku**
7
8     You can try all HTTP operation described in this Swagger spec.
9
10    Find source code of this API [here](https://github.com/mohsen1/petstore-api)
11 host: petstore-api.herokuapp.com
12 basePath: /pet
13 schemes:
14   - http
15   - https
16 consumes:
17   - application/json
18   - text/xml
19 produces:
20   - application/json
21   - text/html
22   - pipo/rire
23 paths:
24   /:
25     get:
26       parameters:
27         - name: limit
28           in: query
29           description: number of pets to return
30           type: integer
31           default: 11
32           minimum: 11
33           maximum: 10000
34       responses:
35         200:
36           description: List all pets
37           schema:
38             title: Pets
39             type: array
40             items:
41               $ref: '#/definitions/Pet'
42     post:
43       parameters:
44         - name: pet
45           in: body
46           description: The pet JSON you want to post
47       schema:
```

PetStore on Heroku ^{1.0.0}

[Base url: petstore-api.herokuapp.com/pet]

This example has a working backend hosted in Heroku

You can try all HTTP operation described in this Swagger spec.

Find source code of this API [here](#)

Schemes

HTTP

default

- GET /
- POST /
- PUT /
- GET /{petId}

Models

Pet > {...} ←

Swagger Codegen

- <https://github.com/swagger-api/swagger-codegen>
- Generate docs, clients and servers project seeds for multiple languages and frameworks.
- Offline
 - Java/Maven, Docker ?
- Online (REST API)
 - Swagger UI <https://generator.swagger.io/>
 - (Fast) Demo via Swagger Editor <http://editor.swagger.io/>

Swagger Utils

- <https://github.com/Swagger2Markup>
- <https://www.npmjs.com/package/yamljs>
- <https://www.npmjs.com/package/bootprint-swagger>

DTO

Data Transfert Object

- Formats
 - JSON, XML, YAML ...
- Frameworks
 - MapStruct.org (Java)

RESTFuL Utils

- Mock
 -
- Test
 - REST Assured
 - SOAP UI
 - Postman
- Docs
 -
- Clients
 - Postman
- Recording and Replay
 - ???
- Load injection
 - JMeter, Gatling
- CLI
 -

Testing RESTFul Services

- REST Assured
 - <http://rest-assured.io/>
 - <https://github.com/rest-assured/rest-assured/wiki/Usage>

Example

```
@Test public void
lotto_resource_returns_200_with_expected_id_and_winners() {
    when().
        get("/lotto/{id}", 5).
    then().
        statusCode(200).
        body("lotto.lottoid", equalTo(5),
            "lotto.winners.winnerId", containsOnly(23, 54));
}
```

SOAP UI

<https://www.soapui.org>

The screenshot shows the SOAP UI interface. On the left is the Project Explorer showing a tree view of the project structure. The main area displays the 'TestSuites' view for the 'Sample SOAP Project Core' project. Below the TestSuites list, there are sections for 'Setup Script' and 'TearDown Script', and a 'TestSuite Log' button. At the bottom left, there are 'Project Properties' and 'Custom Properties' tabs with a table of properties.

Property	Value
Resource Root	
Cache Definitions	true
Project Password	

The screenshot shows the 'TestCases' view for the 'Simple TestSuite'. It displays a list of test cases, each with a green progress bar and the status 'FINISHED'. Below the list, there are tabs for 'Description', 'Properties', 'Setup Script', and 'TearDown Script'. The 'Description' tab is active, showing a log of test steps and their execution times.

TestCases

- Simple Login and Logout w. Properties Steps FINISHED
- Simple Login and Logout Property Expansion FINISHED
- Simple Login and Logout and Login Again FINISHED
- Simple Search TestCase

Step 6 [Test Request - search] OK: took 67 ms

TestCase [Simple Search TestCase] finished with status [FINISHED], time taken = 115

Step 7 [Test Request - logout] OK: took 17 ms

Frontends & Backends

- Tendence architecture
 - Frontend SPA (Single Page Application)
 - AngularJS, React, Vue.js ...
 - Backend RESTFul
 - Remark : services Pub-Sub (websocket, PubNub, ...)
- Frameworks RESTFul
 - JAX-RS implementations
 - Springfox (Spring MVC)
 - Dropwizard
 - Node.JS (Mongoose, ...)
- API Gateway
- OSS
 - Netflix OSS, Consul, ...

JAX-RS

Springfox

Reactive SSE

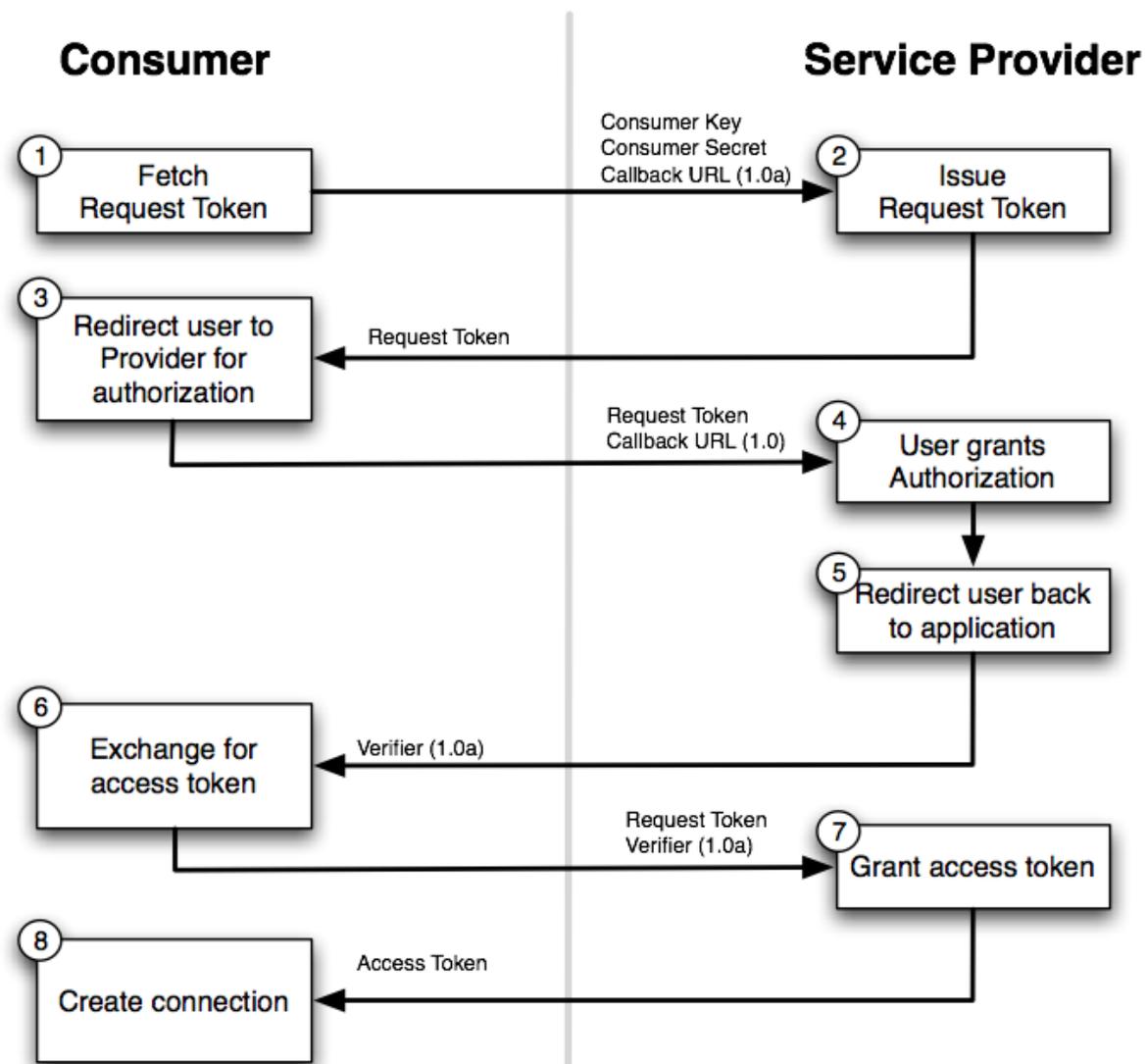
<https://dzone.com/articles/spring-boot-server-sent-events-tutorial>

```
• @RestController
•
• @RequestMapping("/stock/transaction")
•
• class StockTransactionController {
•
•     @Autowired
•
•     StockTransactionService stockTransactionService;
•
•     @GetMapping(produces = MediaType.APPLICATION_STREAM_JSON_VALUE)
•
•     public Flux<StockTransaction> stockTransactionEvents(){
•
•         return stockTransactionService.getStockTransactions();
•
•     }
•
• }
```

Securité

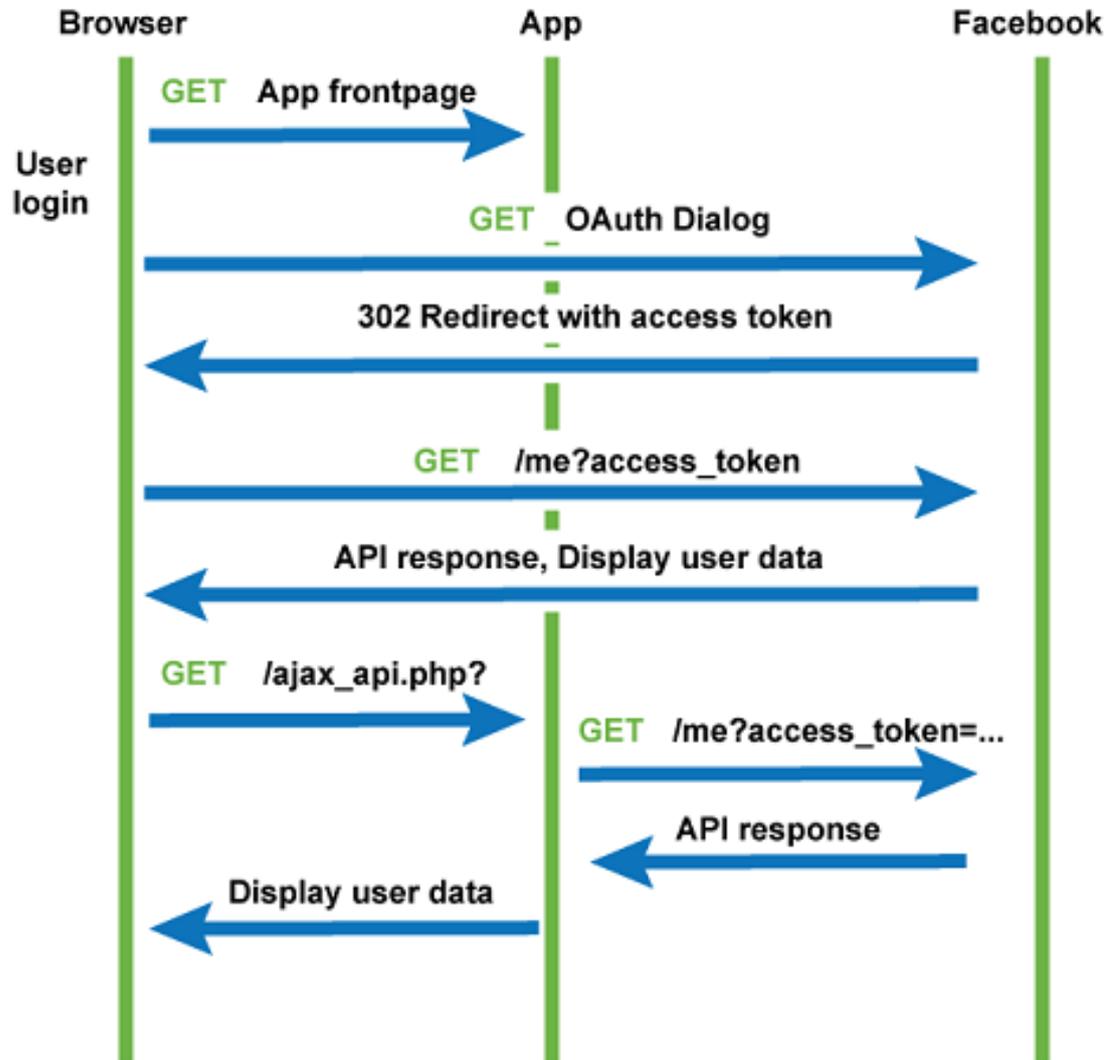
- Confidentialité
 - HTTP over TLS/SSL
- Authentication
 - Basic
 - OAuth 2.0
 - JWT (JSON Web Token)

OAuth 1.0

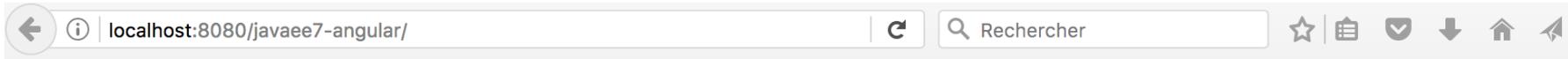


OAuth 2.0

Voir VT2017



Demo Angular + JAX-RS + Swagger



Java EE - Angular Application

List Persons

Id	Name	Description	
1	Uzumaki Naruto	Konoha	✘
2	Hatake Kakashi	Konoha	✘
3	Haruno Sakura	Konoha	✘
4	Uchiha Sasuke	Missing-nin	✘
5	Gaara	Sunagakure	✘
6	Killer Bee	Kumogakure	✘
7	Jiraya	Konoha	✘
8	Namikaze Min...	Konoha	✘
9	Uchiha Madara	Missing-nin	✘
10	Senju Hashirama	Konoha	✘

First Previous **1** 2 3 Next Last

Edit Person

Name:

Description:

Image URL:



Clear

Save

Demo JHipster Spring + Angular 2

- Gestion de courses sportives
- Gestion de bogues (Bug tracker)
- Site de eCommerce

JHipster



👁 Watch ▾

585

★ Star

7,673

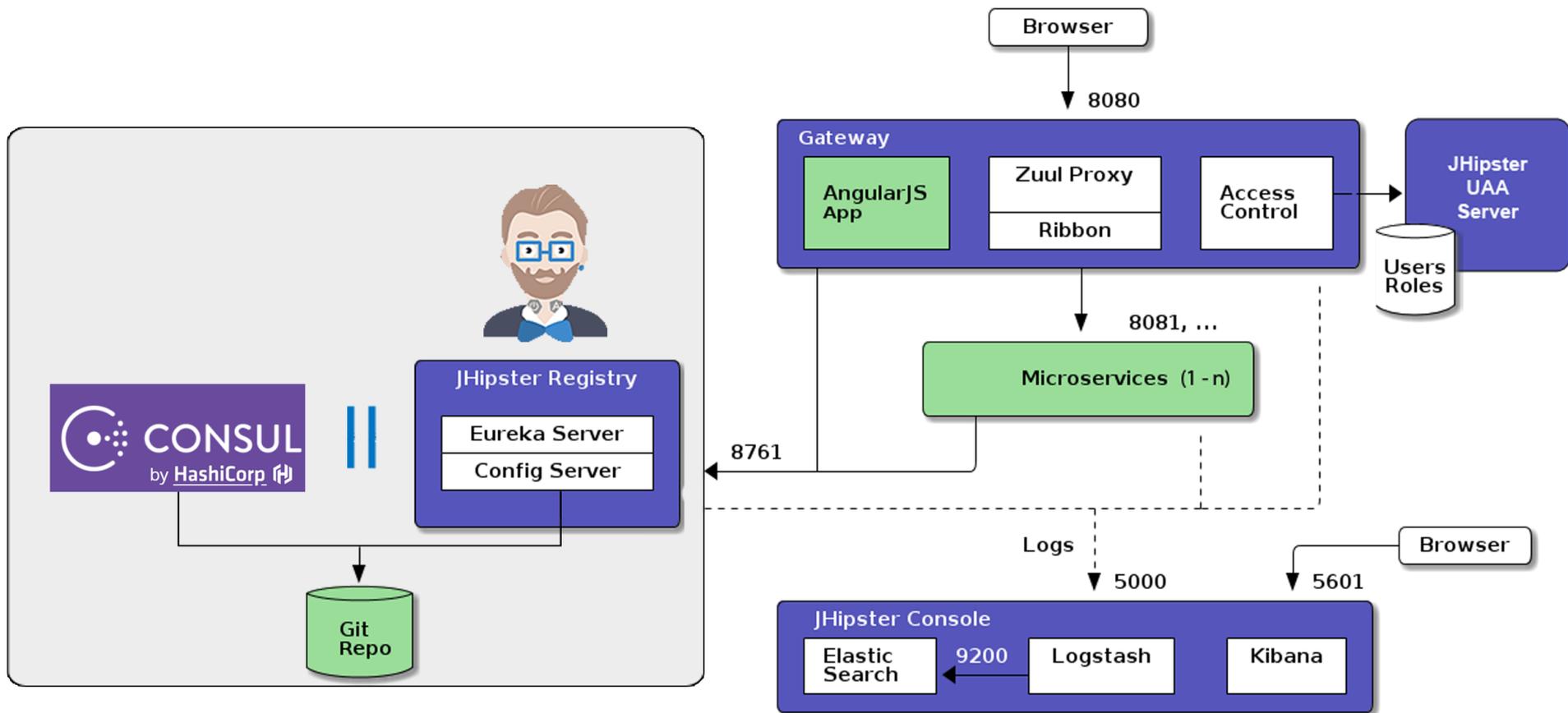
🍴 Fork

1,740

- <https://jhipster.github.io/tech-stack/>
- Single Web page application:
 - AngularJS v1.x or 4
 - Responsive Web Design with Twitter Bootstrap
 - HTML5 Boilerplate
 - Responsive Web Design with Twitter Bootstrap
 - Spring
 - Optional WebSocket support with Spring Websocket
 - Authentication, API Gateway, Rate Limit, Metrics, Kafka, Consul
- Demo
 - <https://jhipster.github.io/showcase/>
 - <https://github.com/mraible/21-points>
 - <http://www.21-points.com>

Example : The JHipster Mini-Book 2.0

NETFLIX | OSS +  +  docker



 elastic +  logstash +  kibana

Netflix OSS

- TODO

Consul

- TODO

REST and IoT

- CoAP (IETF CoRE)
- OneM2M/SmartM2M

Constrained Application Protocol (CoAP)

- Motivation : HTTP/REST-like protocol for constrained sensors
 - Typical configuration : 128KB FlashRAM and 4KB RAM
 - Battery consumption (sleep and periodical wakeup)
- Interaction
 - Request-response
 - Subscribe-Notify
- REST (CRUD)
 - Map on HTTP methods : PUT, GET, POST, DELETE
- Resource discovery
 - /profile URI multicast + DISCOVER method
- Protocol binding
 - UDP and UDP Multicast (16-bit sequence number for reliability)
 - Optionally TCP without “stop and wait”
- Caching
 - Important since sleeping mode
 - CoAP proxy (for subscription ...)
- Standardization
 - IETF Constrained RESTful Environments (CoRE) Working Group

Demo CoAP

- Eclipse Californium
- Plugin Firefox : Copper (Cu)
- Demo
 - <https://github.com/donsez/californium>