API REST

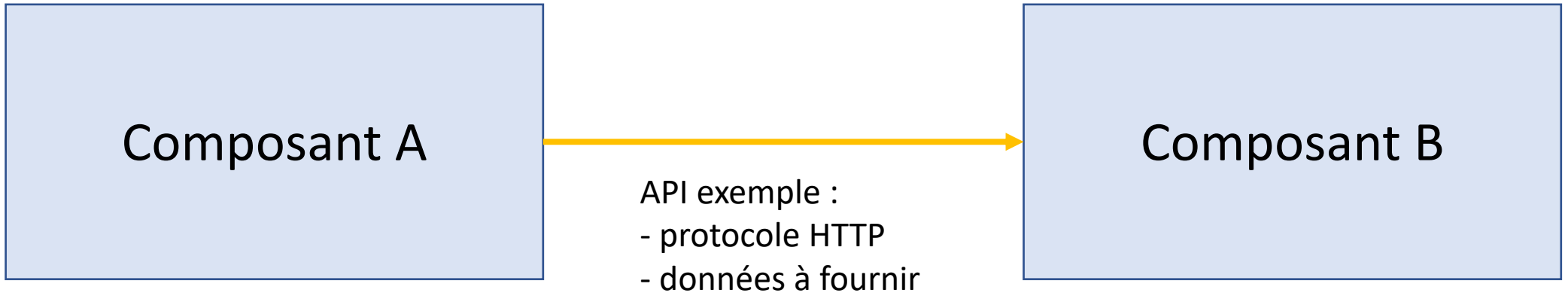
Sommaire

1. Pourquoi une API
2. Pourquoi une API REST
3. RESTful opérations
4. Quelques règles
5. Documenter son API

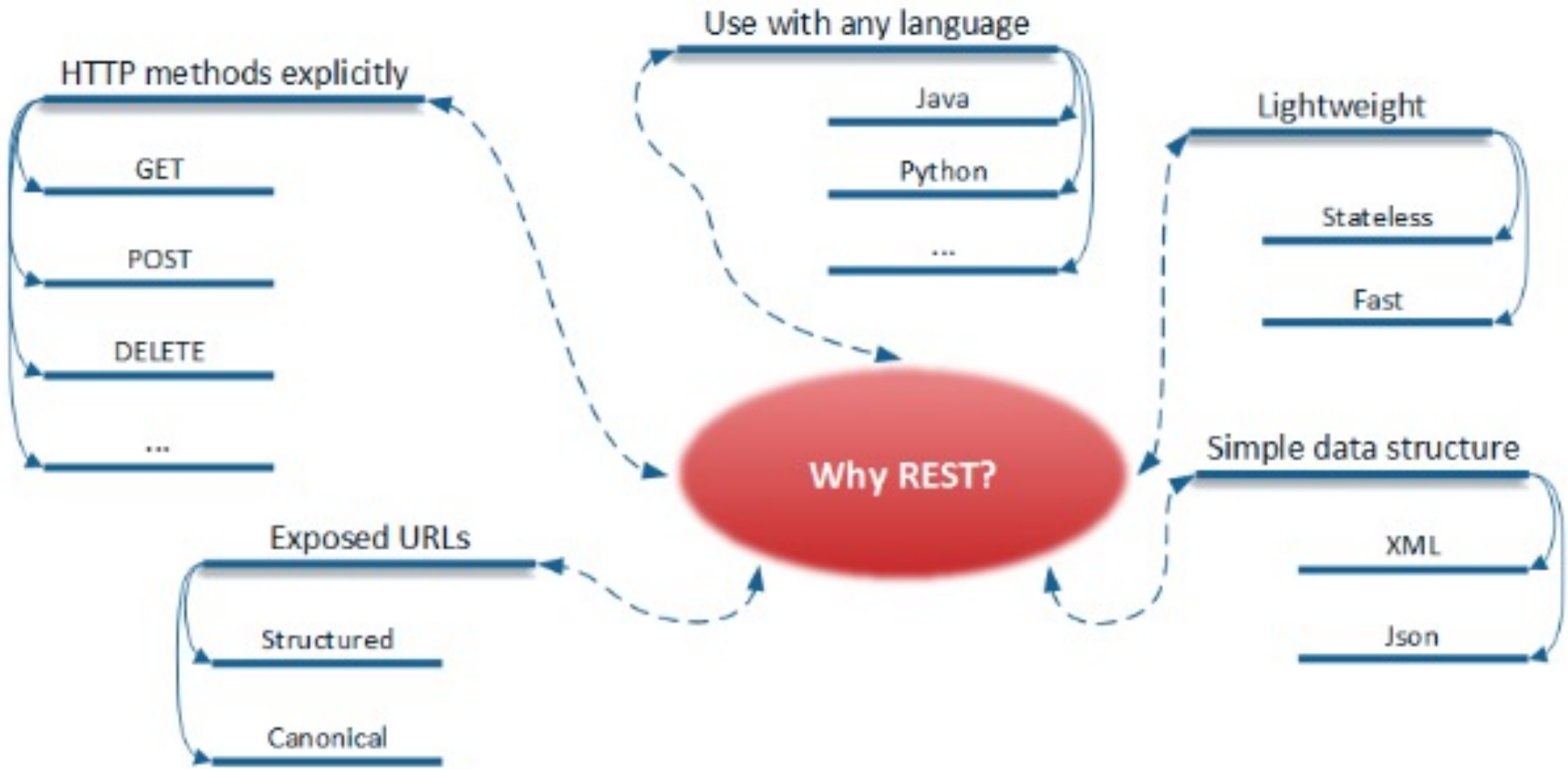
1. Pourquoi une API ?

Définition **A**pplication **P**rogramming **I**nterface

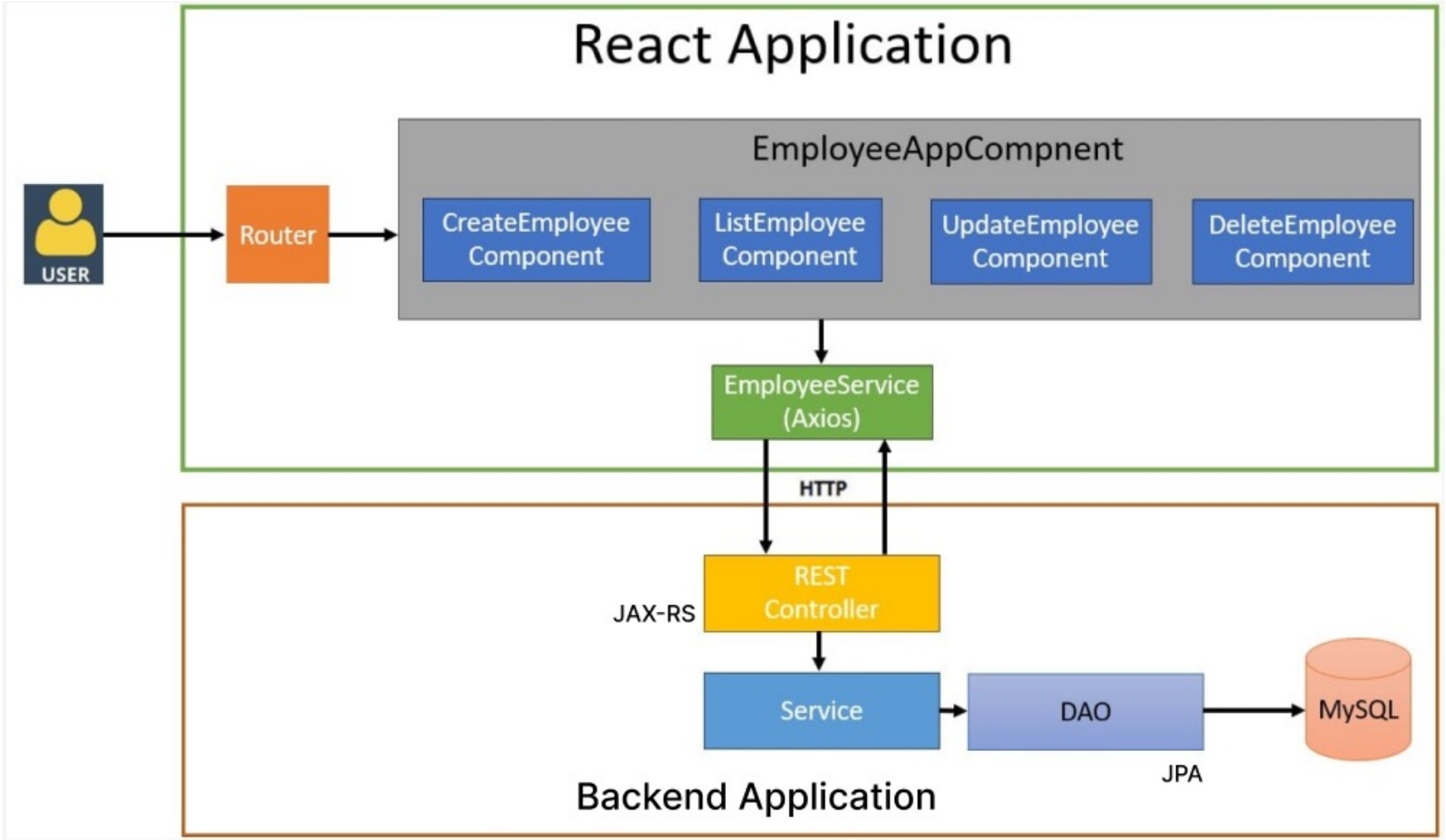
Une API **sert d'intermédiaire** pour permettre à différentes applications logicielles de communiquer entre elles. Elle définit les méthodes et les protocoles que les développeurs peuvent utiliser pour interagir avec un composant logiciel, un service ou une plateforme



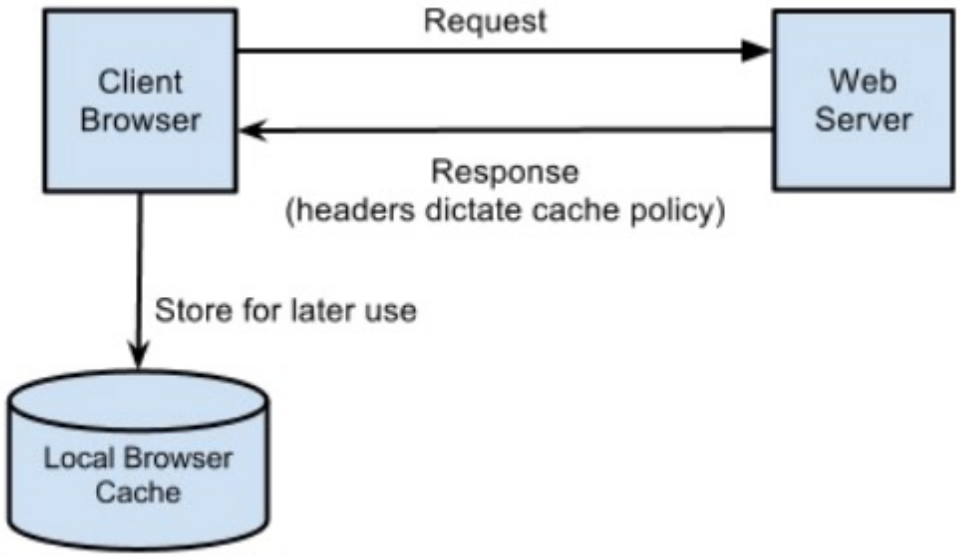
2. Pourquoi une API Rest ?


Use with any language
Java
Python
...
HTTP methods explicitly
GET
POST
DELETE
...
Lightweight
Stateless
Fast
Why REST?
Simple data structure
XML
Json
Exposed URLs
Structured
Canonical

Système en couche

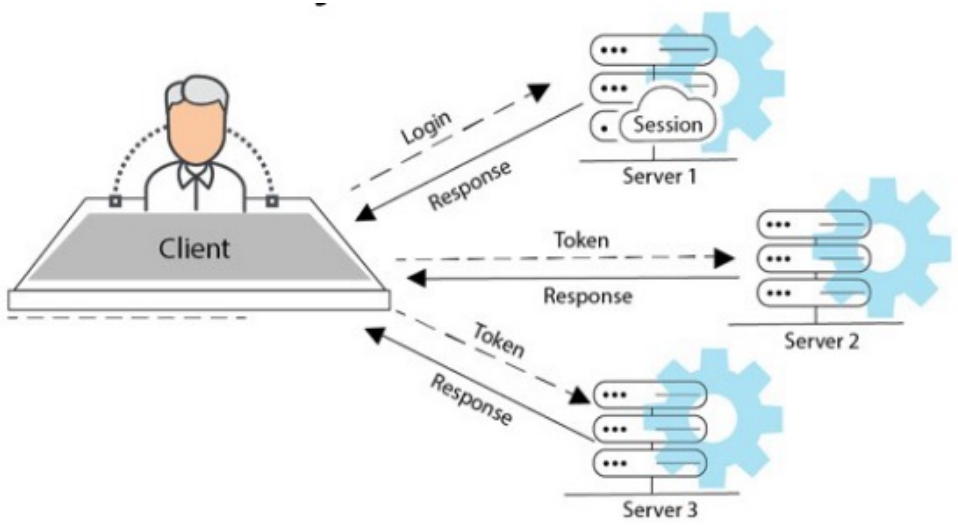


Cache



- Si deux fois la même requête pas besoin de recontacter le serveur

Stateless



- Chaque requête contient toutes les informations nécessaires; pas de session
- => on peut changer de serveur

Histoire

API SOAP : permettant la communication entre des systèmes distribués sur le web.
Cette époque a également vu l'introduction de XML-RPC.

API REST : s'est imposé comme un style architectural pour la conception d'applications en réseau. Les API RESTful, qui utilisent le protocole HTTP (GET, POST, PUT, DELETE), ont été largement adoptées en raison de leur simplicité et de leur évolutivité.

OpenAPI spécification (swagger) : essayer de fournir une documentation standart pour les API REST

GraphQL : une alternative aux API REST

3. RESTFul opérations

Protocole HTTP

Ce base sur le protocole réseau HTTP

Utilise les méthode CRUD du protocole HTTP

- GET retourner une ressource
- POST créer une ressource
- PUT mettre à jour une ressource
- DELETE supprimer une ressource

Sr. No.	API Name	HTTP Method	Path	Status Code	Description
(1)	GET Users	GET	`/api/v1/users`	200 (OK)	All User resources are fetched.
(2)	POST User	POST	`/api/v1/users`	201 (Created)	A new User resource is created.
(3)	GET User	GET	`/api/v1/users/{id}`	200 (OK)	One User resource is fetched.
(4)	PUT User	PUT	`/api/v1/users/{id}`	200 (OK)	User resource is updated.
(5)	DELETE User	DELETE	`/api/v1/users/{id}`	204 (No Content)	User resource is deleted.

Protocole HTTP : retour

Ce base sur le protocole réseau HTTP

Utiliser les codes d'état HTTP appropriés pour indiquer l'état d'une demande :

- 200 OK : La demande a abouti.
- 201 Created : La ressource a été créée avec succès.
- 404 Not found : La ressource n'a pas été trouvée.
- 400 Bad Request : Format de requête non valide.
- 401 Unauthorized : Accès non autorisé.
- 500 Internal Server Error : Erreur côté serveur, etc.

- **Si quelque chose alors format JSON**

1xx Informational

100 Continue
101 Switching Protocols
102 Processing (WebDAV)

2xx Success

★ 200 OK
★ 201 Created
202 Accepted
203 Non-Authoritative Information
★ 204 No Content
205 Reset Content
206 Partial Content
207 Multi-Status (WebDAV)
208 Already Reported (WebDAV)
226 IM Used

3xx Redirection

300 Multiple Choices
301 Moved Permanently
302 Found
303 See Other
★ 304 Not Modified
305 Use Proxy
306 (Unused)
307 Temporary Redirect
308 Permanent Redirect (experiemental)

4xx Client Error

★ 400 Bad Request
★ 401 Unauthorized
402 Payment Required
★ 403 Forbidden
★ 404 Not Found
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Required
408 Request Timeout
★ 409 Conflict
410 Gone
411 Length Required
412 Precondition Failed
413 Request Entity Too Large
414 Request-URI Too Long
415 Unsupported Media Type
416 Requested Range Not Satisfiable
417 Expectation Failed
418 I'm a teapot (RFC 2324)
420 Enhance Your Calm (Twitter)
422 Unprocessable Entity (WebDAV)
423 Locked (WebDAV)
424 Failed Dependency (WebDAV)
425 Reserved for WebDAV
426 Upgrade Required
428 Precondition Required
429 Too Many Requests
431 Request Header Fields Too Large
444 No Response (Nginx)
449 Retry With (Microsoft)
450 Blocked by Windows Parental Controls (Microsoft)
451 Unavailable For Legal Reasons
499 Client Closed Request (Nginx)

5xx Server Error

★ 500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported
506 Variant Also Negotiates (Experimental)
507 Insufficient Storage (WebDAV)
508 Loop Detected (WebDAV)
509 Bandwidth Limit Exceeded (Apache)
510 Not Extended
511 Network Authentication Required
598 Network read timeout error
599 Network connect timeout error

A quoi ça ressemble ?

Question :

« récupérer l'ensemble des étudiants de l'iut »

« récupérer l'étudiant ayant l'id 1 »

« récupérer l'ensemble des étudiants de 3ème année »

« récupérer l'ensemble des étudiants de 3ème année informatique »

« mettre à jour l'ensemble des étudiants de 3ème année informatique nommés toto avec un nouveau nom titi »

A quoi ça ressemble ?

- GET /iut/v1/etudiants
- GET /iut/v1/etudiants/1
- GET /iut/v1/edutiants?annee=3
- GET /iut/v1/edutiants?annee=3&filiere=informatique (filtre)

- PUT /iut/v1/edutiants/ + corps de requête en json

```
{
   name: toto,
   annee: 3
   filiere: informatique,
   nouveauNom: titi
}
```

4. Quelques règles

Bonnes pratiques

- URLs doivent inclure des noms et non des verbes
  - Api/v1/~~getEtudiant~~ --> api/v1/etudiant
  - Api/v1/updateEtudiant --> api/v1/etudiant

Question :

Hum … mais c'est pareil ? Comment faire la différence ?

- Utiliser des nom au pluriel
  - api/v1/etudiant**s**

- Utiliser les statuts de réponse HTTP
  - 200, 505, etc …

Versionnage

L’avantage est que nous pouvons travailler sur de nouvelles fonctionnalités ou améliorations d'une nouvelle version pendant que les clients utilisent toujours la version actuelle et ne sont pas affectés par des changements radicaux.

- /api/**v1**/etudiants
- /api/**v2**/etudiants

Profondeur maximale

Eviter les urls plus profonde que
`resource/identifier/resource`.

All articles in (or belonging to) this magazine:

```
GET /api/v1/magazines/1234/articles.json HTTP/1.1
Host: www.example.gov.au
Accept: application/json, text/javascript
```

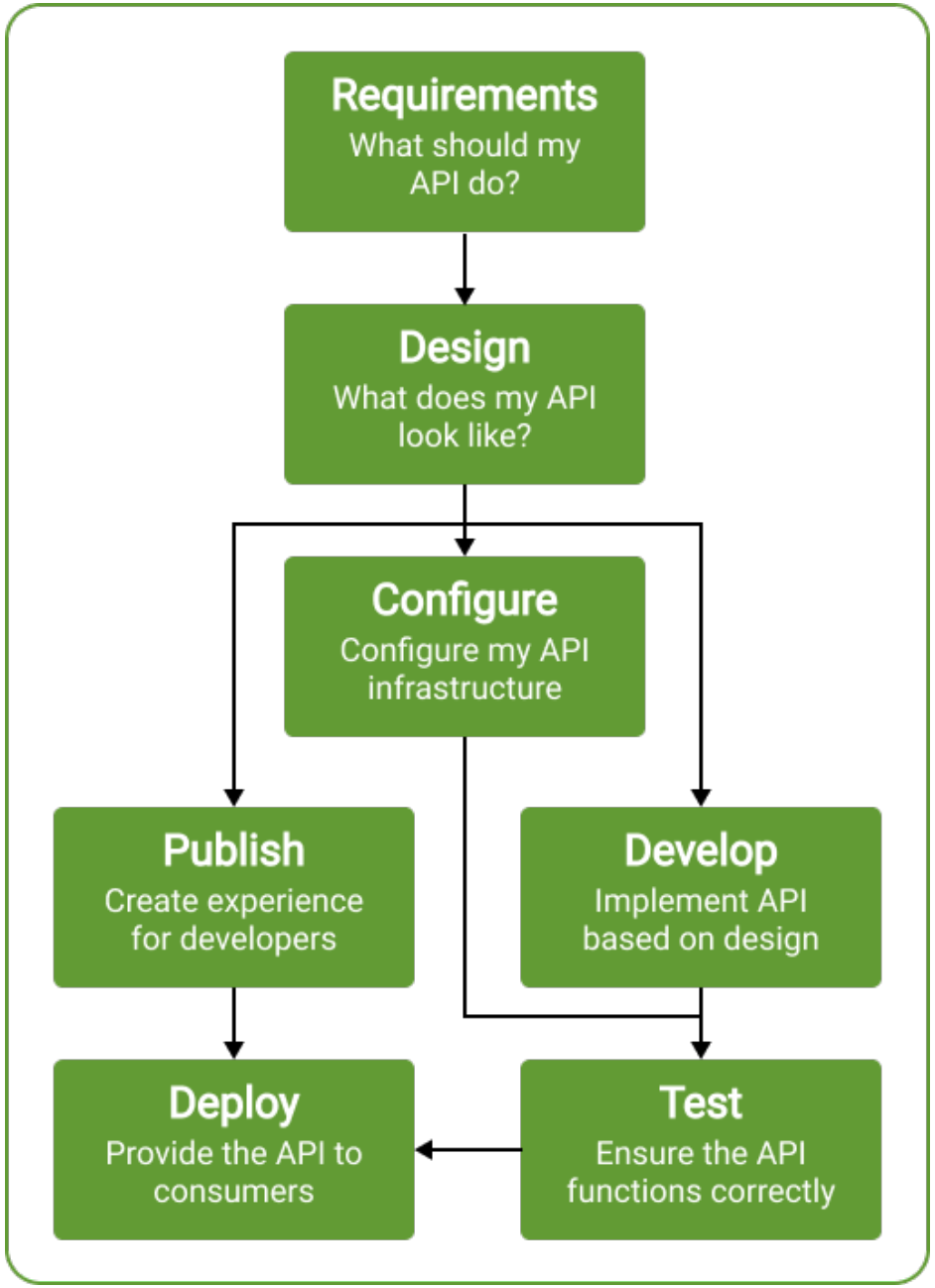
Si une ressource ne peut pas exister sans une autre alors sous-ressource
GET /factures/1/details

Le détail de la facture ne peut pas exister sans la facture (en-tête, etc)

5. Documenter API REST

OpenAPI

The OpenAPI Specifications provide a formal standard for describing HTTP APIs.

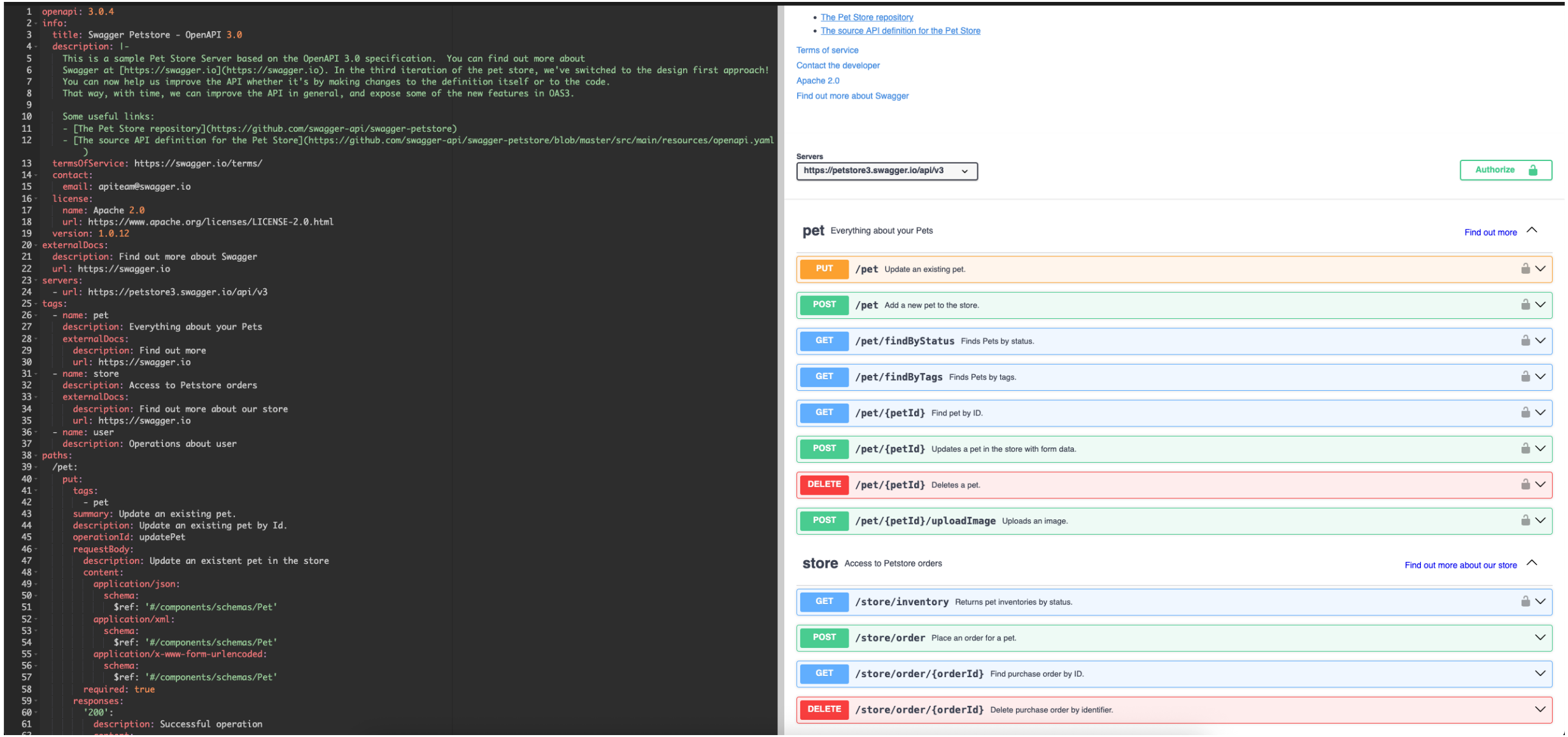


API First approche

JSON


```
{
  "openapi": "3.0.0",
  "info": {
    "title": "Simple API overview",
    "version": "2.0.0"
  },
  "paths": {
    "/": {
      "get": {
        "operationId": "listVersionsv2",
        "summary": "List API versions",
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "examples": {
                  "foo": {
                    "value": {
                      "versions": [
                        {
                          "status": "CURRENT",
                          "updated": "2011-01-21T11:33:21Z",
```


Génère le code Java pour les endpoints



```yaml
openapi: 3.0.4
info:
  title: Swagger Petstore - OpenAPI 3.0
  description: |-
    This is a sample Pet Store Server based on the OpenAPI 3.0 specification.  You can find out more about
    Swagger at [https://swagger.io](https://swagger.io). In the third iteration of the pet store, we've switched to the design first approach!
    You can now help us improve the API whether it's by making changes to the definition itself or to the code.
    That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

    Some useful links:
    - [The Pet Store repository](https://github.com/swagger-api/swagger-petstore)
    - [The source API definition for the Pet Store](https://github.com/swagger-api/swagger-petstore/blob/master/src/main/resources/openapi.yaml
      )
  termsOfService: https://swagger.io/terms/
  contact:
    email: apiteam@swagger.io
  license:
    name: Apache 2.0
    url: https://www.apache.org/licenses/LICENSE-2.0.html
  version: 1.0.12
externalDocs:
  description: Find out more about Swagger
  url: https://swagger.io
servers:
  - url: https://petstore3.swagger.io/api/v3
tags:
  - name: pet
    description: Everything about your Pets
    externalDocs:
      description: Find out more
      url: https://swagger.io
  - name: store
    description: Access to Petstore orders
    externalDocs:
      description: Find out more about our store
      url: https://swagger.io
  - name: user
    description: Operations about user
paths:
  /pet:
    put:
      tags:
        - pet
      summary: Update an existing pet.
      description: Update an existing pet by Id.
      operationId: updatePet
      requestBody:
        description: Update an existent pet in the store
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Pet'
          application/xml:
            schema:
              $ref: '#/components/schemas/Pet'
          application/x-www-form-urlencoded:
            schema:
              $ref: '#/components/schemas/Pet'
        required: true
      responses:
        '200':
          description: Successful operation
```