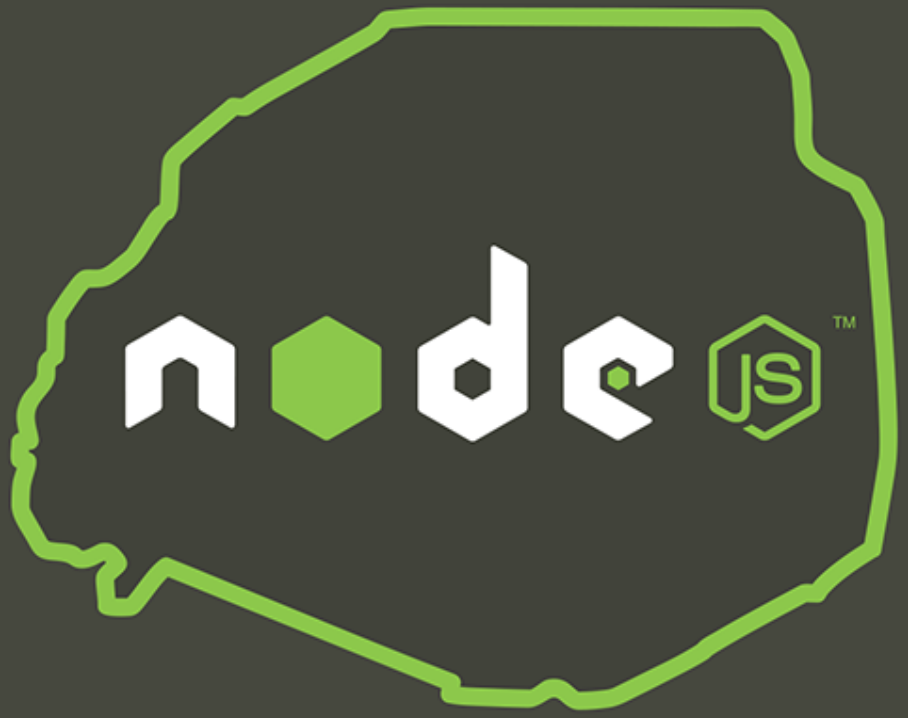

REST World

REST -> NIVEAU 0

```
To: <soap@example.org>
From: <soap@client.com>
Reply-To: <soap@client.com>
Date: Tue, 15 Nov 2001 23:27:00 -0700
Message-Id: <1F75D4D515C3EC3F34FEAB51237675B5@client.com>
MIME-Version: 1.0
Content-Type: text/xml; charset=utf-8
Content-Transfer-Encoding: QUOTED-PRINTABLE

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<m:echoString xmlns:m="http://soapinterop.org/">
<inputString>get your SOAP over SMTP here !</inputString>
</m:echoString>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP via SMTP

REST -> NIVEAU 0

UTILISATION DU PROTOCOLE HTTP

Ce niveau est "facultatif" (mais on a pas mieux aujourd'hui)

UTILISATION DU PROTOCOLE HTTP

Pourquoi ?

- Pour pouvoir valider les niveaux suivants ...
- HTTP propose des méthodes HTTP, des headers HTTP et des codes HTTP bien pratiques
- HTTP est client-serveur
- HTTP est uniforme du fait de son universalité sur le web
- HTTP s'utilise sur de nombreux supports, dont les mobiles
- HTTP est performant (pour un protocole textuel) ... merci SPDY

```
POST /schema-instance HTTP/1.1
Host: www.soap.example.org
SOAPAction: "Some-URI"

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/enco
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<m:echoString xmlns:m="http://soapinterop.org/">
<inputString>get your SOAP over HTTP here !</inputString>
</m:echoString>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

...
```

SOAP via HTTP

LEVEL UP !

REST -> NIVEAU 1

```
POST /schema-instance HTTP/1.1
Host: www.soap.example.org
SOAPAction: "Some-URI"

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/enco
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<m:echoString xmlns:m="http://soapinterop.org/">
<inputString>get your SOAP over HTTP here !</inputString>
</m:echoString>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

...
```

SOAP via HTTP

```
GET /data/2.0/Equipements/get_equipements HTTP/1.1
Host: api.paris.fr

HTTP/1.1 200 OK

{
  "status": "success",
  "data": [
    {
      "name": "Crèche collective municipale de Cotte",
      "address": "7 rue de Cotte ",
      "zipCode": 75012
    }
  ],
  "api-version": "2.0",
  // ...
}
```

```
GET /data/2.1/Equipements/get_equipements HTTP/1.1
Host: api.paris.fr

HTTP/1.1 200 OK

{
  "status": "success",
  "data": [
    {
      "name": "Crèche collective municipale de Cotte",
      "address": "7 rue de Cotte ",
      "zipCode": 75012
    }
  ],
  "api-version": "2.1",
  // ...
}
```

1 ressource, 2 URIs

```
GET /2.0/lists/member.json HTTP/1.1
Host: us2.api.mailchimp.com

HTTP/1.1 200 OK

{
  "apikey": "example apikey",
  "id": "example id",
  "email": {
    "email": "example email",
    "euid": "example euid",
    "leid": "example leid"
  },
  // ...
}
```

```
GET /2.0/lists/member.xml HTTP/1.1
Host: us2.api.mailchimp.com

HTTP/1.1 200 OK

<xml>
  <apikey>example apikey</apikey>
  <id>example id</id>
  <email>
    <email>example email</email>
    <euid>example euid</euid>
    <leid>example leid</leid>
  </email>
  <!-- ... -->
</xml>
```

1 ressource, 2 URIs

REST -> NIVEAU 1

1 RESOURCE = 1 URI

1 RESOURCE = 1 URI

Pourquoi ?

- Pour pouvoir utiliser la puissance des URI (Uniform Resource Identifiers)
- URI c'est uniforme du fait de son universalité
- 1 URI, donc pas d'état dans l'adresse
- 1 URI, c'est plus facile à entretenir, tous les chemins vont à Rome

COOL URIS DON'T CHANGE

Utilisez des noms, pas des verbes

Une liste de clients ? /client
Un client particulier ? /client/1

OU

Une liste de clients ? /clients
Un client particulier ? /clients/1

Important : la consistance des noms

COOL URIS DON'T CHANGE 2

Une liste de clients ?
/client : ressource à accéder
/client/ : redirection (301)

OU

l'inverse

COOL URIS DON'T CHANGE 3

Pour les noms composés, évitez les Majuscules, préférez les tirets (-)

Une liste de clients techniques ?

NON : /technicalClient
OUI : /technical-client

UPRI

Pour un client unique, un identifiant statique et unique (un ID)
MAIS, aussi des termes lisibles statiques et uniques

/technical-client/111855

/technical-client/111855-honda

UPRI : unique-id-plus-redundant-information

URI HACKABLES ?

Hum ... nous verrons ça au niveau 3

GÉRER LES FORMATS ?

Toute l'intelligence doit être stockée dans les paramètres
Pagination, filtre, tri, format, ...

NON : /clients.xml/0/10?sort=name
OUI : /clients?format=xml&sort=name&start=0&size=10

VERSIONNER SON API ?

http://company.com/api/v3.0/customer/123/v2.0/orders/4321/

See you at level 4 !

```
GET data/Equipements/get_equipements HTTP/1.1
Host: api.paris.fr

...
```

HTTP et Une URI = 1 Resource

LEVEL UP !

REST -> NIVEAU 2

```
POST /lists/update-member.json HTTP/1.1
Host: us2.api.mailchimp.com

...
```

```
POST /list/interest-grouping-delete HTTP/1.1
Host: us2.api.mailchimp.com


...
```

Action dans l'URI

REST -> NIVEAU 2

PAS D'ACTIONS DANS L'URL, UTILISEZ LES MÉTHODES HTTP

QUI CONNAIT TOUTES LES MÉTHODES HTTP ?

- GET
- POST
- PUT
- DELETE

Mais les méthodes HTTP, ce n'est pas QUE du CRUD

- HEAD
- OPTIONS
- PATCH
- TRACE
- CONNECT

GET

Read du CRUD

Nullipotent/Nilpotent

Sert à récupérer des données

GET

`/client`
Récupère la collection contenant les URI correspondant à chaque client (ou les données directement)

`/client/1`
Récupère l'entité via ses données représentant le client

Client-serveur => la représentation <> la manière dont elle est persistée

Donnée unitaire + politique de cache adaptée = performance

```
GET /users/vbardales HTTP/1.1
Host: api.github.com

HTTP/1.1 200 OK

{
  "login": "vbardales",
  "id": 1446444,
  "url": "https://api.github.com/users/vbardales",
  "followers_url": "https://api.github.com/users/vbardales/follower
  // ...
}
```

GET d'une entité

```
GET /users/vbardales/followers HTTP/1.1
Host: api.github.com

HTTP/1.1 200 OK

[
  {
    "login": "valdo404",
    "id": 175694,
    "url": "https://api.github.com/users/valdo404",
    "followers_url": "https://api.github.com/users/valdo404/followers"
  },
  {
    "login": "thomaroger",
    "id": 1062137,
    "url": "https://api.github.com/users/thomaroger",
    "followers_url": "https://api.github.com/users/thomaroger/followers"
  },
  // ...
]
```

GET d'une collection

Et comment on cache facilement, hein ?

```
GET /users/vbardales/followers HTTP/1.1
Host: api.github.com

HTTP/1.1 200 OK

[
  "https://api.github.com/users/valdo404",
  "https://api.github.com/users/thomaroger",
  "https://api.github.com/users/jeremymarc"
]
```

GET d'une collection

DELETE

Delete du CRUD

Idempotent

Sert à supprimer des données

DELETE

/client
Vide la collection

/client/1
Supprime l'entité

```
DELETE /repos/vbardales/AdmingeneratorGeneratorBundle HTTP/1.1
Host: api.github.com

HTTP/1.1 204 No Content
```

DELETE d'une entité

PUT

Create et Update du CRUD

Idempotent

Sert à écrire des données (ressource entière)

PUT

`/client`
Ecrase toute la collection de clients pour la remplacer par la nouvelle collection transmise

`/client/1`
Si l'entité existe, l'écrase, sinon, la crée

La ressource entière doit être transmise
Pas de mises à jour partielles

```
PUT /users/vbardales/followers HTTP/1.1
Host: api.github.com

[
  "https://api.github.com/users/valdo404",
  "https://api.github.com/users/jeremymarc"
]


HTTP/1.1 200 OK
...
```

PUT d'une collection

POST

2 comportements

Non-idempotent

POST

Cas 1 : provoque un changement de la ressource

/client/1/status
Si l'entité n'existe pas, la crée mais si elle existe, fait un autre traitement (405 Method Not Allowed)

/client/1/vote/18
Si l'entité existe et que le client n'a pas encore déposé d'entité 'vote', ajoute un vote, sinon, retire le vote du client

Pas fait pour mettre à jour une ressource

POST

Cas 2 : ajoute une entité à une collection sans précision de son futur identifiant

/client/1/invoice
Ajoute une entité 'invoice' à la collection de factures du client

```
POST /users/vbardales/followers HTTP/1.1
Host: api.github.com

https://api.github.com/users/thomaroger

HTTP/1.1 204 No Content
...
```

POST d'une nouvelle entité sur une collection

PATCH

Idempotent

Sert à mettre partiellement à jour des données

RFC 6902 - JavaScript Object Notation (JSON) Patch

```
PATCH /my/data/123 HTTP/1.1
Host: example.org
Content-Type: application/json-patch+json

[
  { "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" }
]

HTTP/1.1 200 OK
...
```

```
GET /my/data/123 HTTP/1.1
Host: example.org

HTTP/1.1 200 OK

{
  "a": {
    "b": {
      "d": 42,
      "e": 42,
    }
  }
}
```

PATCH d'une ressource

OPTIONS

Nullipotent

Retourne les méthodes supportées pour une URI donnée

```
OPTIONS /my/data HTTP/1.1
Host: example.org

HTTP/1.1 204 No Content
Allow: HEAD,GET,PUT,DELETE,OPTIONS
```

OPTIONS sur une ressource

HEAD

Nullipotent

L'idée originale est :
Retourne l'entête du message obtenu lors d'un GET sans le corps de la requête

Moi je préfère :
Retourne un entête de message sans corps contenant certaines des informations obtenues lors d'un GET

```
GET /my/data?start=3&size=4 HTTP/1.1
Host: example.org

206 Partial Content

[
  "test4",
  "test5",
  "test6",
  "test7"
]
```

```
HEAD /my/data?start=3&size=4 HTTP/1.1
Host: example.org

HTTP/1.1 200 OK
Accept-Ranges: data
Content-Range: data 3-7/18
```

HEAD sur une ressource

HEAD

Vous en voulez plus ?

See you at level 3 !

CAS PARTICULIER :

```
POST /me?method=put HTTP/1.1
Host: api.viadeo.com

...
```

Paramètre `method` : pour les supports ne gérant pas toute la norme HTTP 1.1

```
POST /schema-instance HTTP/1.1
Host: www.soap.example.org
Content-Type: application/soap+xml
SOAPAction: "Some-URI"

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/enco
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
  <m:GetStockPrice xmlns:m="http://www.example.org/stock">
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

...
```

SOAP ne suit pas RESTv2 (utilisation abusive de POST)

LEVEL UP !

REST -> NIVEAU 3

```
GET /users/vbardales HTTP/1.1
Host: api.github.com

HTTP/1.1 200 OK

{
  "login": "vbardales",
  "id": 1446444,
  "url": "https://api.github.com/users/vbardales",
  "followers_url": "https://api.github.com/users/vbardales/follower
  // ...
}
```

Compatible RESTv3

REST -> NIVEAU 3

Communément nommé RESTful

HATEOAS

Hypermedia as the Engine of Application State

HATEOAS

Je ne dois en aucun cas avoir à deviner ...
- ... comment accéder à une ressource donnée
- ... quelles sont les relations entre ressources et collections

Self-descriptive : mon index résume l'ensemble des ressources que j'expose et comment y accéder

Je suis sur une ressource, je veux y réaccéder ...

```
GET /cities/paris/events/29 HTTP/1.1
Host: api.humantalks.com

HTTP/1.1 200 OK

<?xml version="1.0" encoding="utf-8" ?>
<event>
  <id>29</id>
  <city>Paris</city>
  <date>2013-04-09T19:00:00Z</date>
  <link rel="self" href="/cities/paris/events/29.xml" />
  <link rel="city" href="/cities/paris/events.xml" />
  <link rel="talks" href="/events/29/talks.xml" />
</event>
```

Un lien `self` existe

Pratique cette balise link ...

OUI, MAIS EN JSON ?

http://stateless.co/hal_specification.html

application/hal+json

```
{
  "_links": {
    "self": { "href": "/orders" },
    "next": { "href": "/orders?page=2" },
  },
  "currentlyProcessing": 14,
  "shippedToday": 20,
  "_embedded": {
    "ea:order": [{
      "_links": {
        "self": { "href": "/orders/123" },
        "ea:basket": { "href": "/baskets/98712" },
        "ea:customer": { "href": "/customers/7809" }
      },
      "total": 30.00,
      "currency": "USD",
      "status": "shipped"
    }]
  }
}
```

Verbeux ...

OUI, MAIS EN JSON ?

Sinon, à la débrouille ...

```
GET /my/user/vbardales HTTP/1.1
Host: example.org

HTTP/1.1 200 OK

{
  "login": "vbardales",
  "_self": "https://api.github.com/users/vbardales",
  "name": "Virginie Bardales"
}
```

Un préfixe peut suffire

Je suis sur une collection paginée ...

```
GET /my/data?start=3&size=4 HTTP/1.1
Host: example.org

200 OK

{
  "_self": "http://example.org/my/data",
  "_collection": [
    "test4",
    "test5",
    "test6",
    "test7"
  ]
}
```

On rajoute un niveau d'indentation

Je suis sur une collection paginée ...

```
GET /my/data?start=3&size=4 HTTP/1.1
Host: example.org

206 Partial Content
Accept-Ranges: data
Content-Range: data 3-7/18

{
  "_self": "http://example.org/my/data",
  "_collection": [
    "test4",
    "test5",
    "test6",
    "test7"
  ]
}
```

Pas mal avec les Headers adaptés

Je suis sur une collection paginée ...

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-

206 Partial Content
Accept-Ranges: data
Content-Range: data 3-7/18

{
  "_self": "http://example.org/my/data",
  "_collection": [
    "test4",
    "test5",
    "test6",
    "test7"
  ]
}
```

Pas mal pour libérer mon URL

Je suis sur une collection paginée ...

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-

206 Partial Content
Accept-Ranges: data
Content-Range: data 3-7/18

{
  "_self": "http://example.org/my/data",
  "_relatedData": "http://example.org/my/related-data",
  "_collection": [
    "test4",
    "test5",
    "test6",
    "test7"
  ]
}
```

Dès que je rajoute d'autres relations ...

Je suis sur une collection paginée ...

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-7

206 Partial Content
Accept-Ranges: data
Content-Range: data 3-7/18

{
  "_self": "http://example.org/my/data?start=3&size=4",
  "_relatedData": "http://example.org/my/related-data",
  "_page": {
    "total": 18,
    "next": "http://example.org/my/data?start=7&size=4",
    "prev": "http://example.org/my/data?start=0&size=4",
    "first": "http://example.org/my/data?start=0&size=4",
    "last": "http://example.org/my/data?start=14&size=4"
  }
  "_collection": [
    "test4",
    "test5",
    "test6",
    "test7"
  ]
}
```

Retour des paramètres d'URL

RFC 5988 - Web Linking

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-7

206 Partial Content
Accept-Ranges: data
Content-Range: data 3-7/18
Link: <http://example.org/my/data?start=3&size=4>; rel="self",
<http://example.org/my/related-data>; rel="relatedData",
<http://example.org/my/data?start=7&size=4>; rel="next",
<http://example.org/my/data?start=0&size=4>; rel="prev",
<http://example.org/my/data?start=0&size=4>; rel="first",
<http://example.org/my/data?start=14&size=4>; rel="last"

[
  "test4",
  "test5",
  "test6",
  "test7"
]
```

Disparition du niveau d'indentation

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-7

206 Partial Content
Accept-Ranges: data
Content-Range: data 3-7/18
Link: <http://example.org/my/data>; rel="self",
<http://example.org/my/related-data>; rel="relatedData"

[
  "test4",
  "test5",
  "test6",
  "test7"
]
```

Suppression des liens inutiles

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-7

206 Partial Content
Accept-Ranges: data
Content-Range: data 3-7/18
Content-Location: http://example.org/my/data
Link: <http://example.org/my/related-data>; rel="relatedData"

[
  "test4",
  "test5",
  "test6",
  "test7"
]
```

Plus de lien `self`

Je crois que maintenant, ma méthode HEAD peut vraiment être utile ...

CANONICAL LINK

```
GET /my/city/19-pekin HTTP/1.1
Host: example.org
Range: data 3-7

200 OK
Content-Location: http://example.org/my/city/19-pekin

...
```

```
GET /my/city/19-beijing HTTP/1.1
Host: example.org
Range: data 3-7

200 OK
Content-Location: http://example.org/my/city/19-beijing
Link: <http://example.org/my/city/19-pekin>; rel="canonical"

...
```

UPRI = lien canonique

```
HEAD /my/data HTTP/1.1
Host: example.org
Range: data 18-

216 Requested Range Not Satisfiable
```

Codes status HTTP

LEVEL UP !

REST -> NIVEAU 4

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-7
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q

...
```

Clients peu aptes à manipuler les Headers HTTP

Donc Headers HTTP peu utilisés

Mais dans le meilleur des mondes ...

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-7
Accept: image/gif, image/jpeg, image/pjpeg, application/x-ms-applic
        application/vnd.ms-xpsdocument, application/xaml+xml,
        application/x-ms-xbap, application/x-shockwave-flash,
        application/x-silverlight-2-b2, application/x-silverlight,
        application/vnd.ms-excel, application/vnd.ms-powerpoint,
        application/msword, */*

...
```

Merci IE

Ne pas abuser des bonnes choses (pas de compression)

Trafic inutile

REST -> NIVEAU 4

LA REVANCHE DES HEADERS

ou la puissance des media-types !

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-7
Accept: application/balloon+json

406 Not Acceptable
```

Accept fantaisiste

VERSIONING D'API

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-7
Accept: application/org.example.data-v2+json

HTTP/1.1 200 OK
Content-Type: application/org.example.data-v2+json

...
```

Définit la version attendue

OUI MAIS ...

Lors d'un POST, je ne peux pas mettre ces headers-là

Oui, mais ... lors d'un POST, dernière version de l'API ?

OUI MAIS ...

Si je ne mets pas de Accept, que se passe-t-il ?

Une erreur HTTP 406 Not Acceptable ?

Par défaut la dernière version de l'API ?

CORS

Explicitement autoriser ces headers "custom"

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-7

206 Partial Content
Accept-Ranges: data
Content-Range: data 3-7/18
Content-Location: http://example.org/my/data
Link: <http://example.org/my/related-data>; rel="relatedData"

[
  "test4",
  "test5",
  "test6",
  "test7"
]
```

API plutôt bien faite

LEVEL UP !

REST -> NIVEAU 5

ET SI ...

... j'ai régulièrement besoin de connaitre la plus faible surface entre mes triangles et mes carrés ?
... je veux savoir dans combien de temps le prochain bus passe ?

JE PEUX ...

... écrire une requête complexe à base de filtres et d'ordres et récupérer le premier résultat
... passer en RESTv5

REST -> NIVEAU 5

RÉINTRODUIRE DU MÉTIER

Approche comportementale

```
GET /my/data HTTP/1.1
Host: example.org
Range: data 3-7

206 Partial Content
Accept-Ranges: data
Content-Range: data 3-7/18
Content-Location: http://example.org/my/data
Link: <http://example.org/my/related-data>; rel="relatedData",
<http://example.org/my/data/related-behavior>; rel="relatedBehavior",

[
  "test4",
  "test5",
  "test6",
  "test7"
]
```

Ajoutons du métier

```
POST /my/data/related-behavior HTTP/1.1
Host: example.org

...
```

Nouvelle dimension à POST

YOU WIN !

LES FRAMEWORKS

EXPRESS FROM SCRATCH

```
var express = require('express');
var db = require('dbInitialization');

var app = express();
// setting up env (app.use(...))

// development only
if ('development' == app.get('env')) {
  app.use(express.errorHandler());
}

app.get('/user', function(req, res) {
  db.collection('user').find().toArray(function (err, items) {
    res.json(items);
  })
});
app.post('/user', function(req, res) { // ... });

// ...

// start server
```

A la main ...

EXPRESS & NODE-RESTFUL & MONGOOSE

https://github.com/baugarten/node-restful

```
var express = require('express'),
    restful = require('node-restful'),
    mongoose = restful.mongoose;
var app = express();

// setting up env (app.use(...))

mongoose.connect("mongodb://localhost/resources");

var Resource = app.resource = restful.model('resource', mongoose.Schema({
    title: 'string',
    year: 'number',
  }))
  .methods(['get', 'post', 'put', 'delete']);

Resource.register(app, '/resources');

// start server
```

Pas RESTful ...

FLATIRON'S RESTFUL

https://github.com/flatiron/restful

```
var flatiron = require('flatiron'),
    fixtures = require('../test/fixtures'),
    restful = require('../lib/restful'),
    resourceful = require('resourceful');

var app = module.exports = flatiron.app;

app.resources = {};
app.resources.Creature = fixtures.Creature;
app.resources.Album = fixtures.Album;

app.use(flatiron.plugins.http, {
  headers: {
    'x-powered-by': 'flatiron ' + flatiron.version
  }
});
app.use(restful);

// start server
```

Pas RESTful ...

SPUMKO'S HAPI

https://github.com/spumko/hapi

```
var Hapi = require('hapi');
var db = require('dbInitialization');

// Create a server with a host and port
var server = Hapi.createServer('localhost', 8000);

// Add the route
server.route({
    method: 'GET',
    path: '/user',
    handler: function (request, reply) {
      db.collection('user').find().toArray(function (err, items) {
        reply(items);
      });
    }
});
// ...

// start server
```

Pas de génération automatique

KOA & KOA-ROUTE

http://koajs.com

```
var koa = require('koa');
var route = require('koa-route');
var db = require('dbInitialization');

var app = koa();
// setting up env (app.use(...))

app.use(route.get(function *() {
  this.body = (yield db.collection('user').find()).toArray();
});

// start server
```

<3 ES6