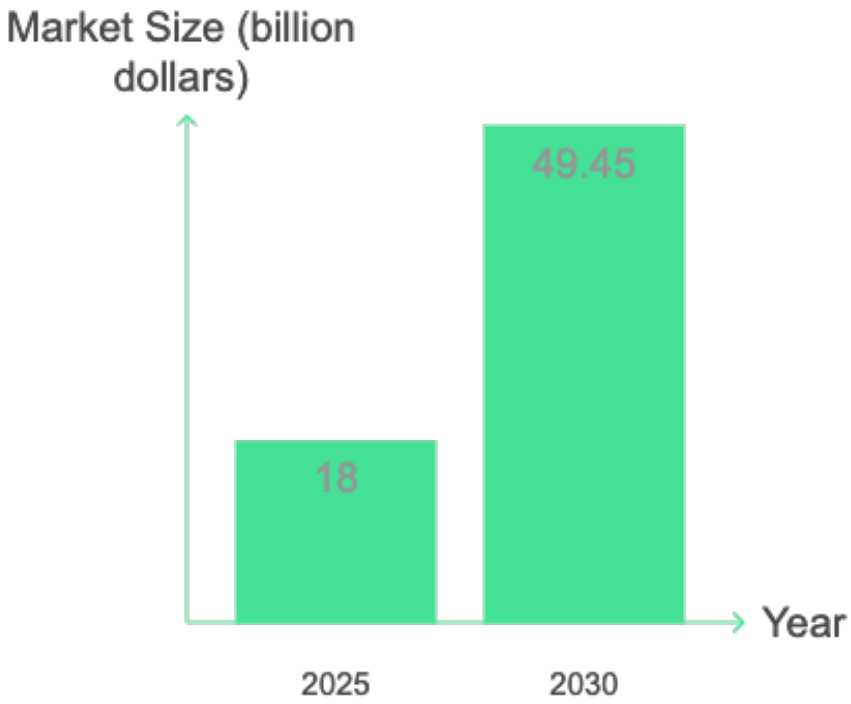
Conception d’une API RESTful

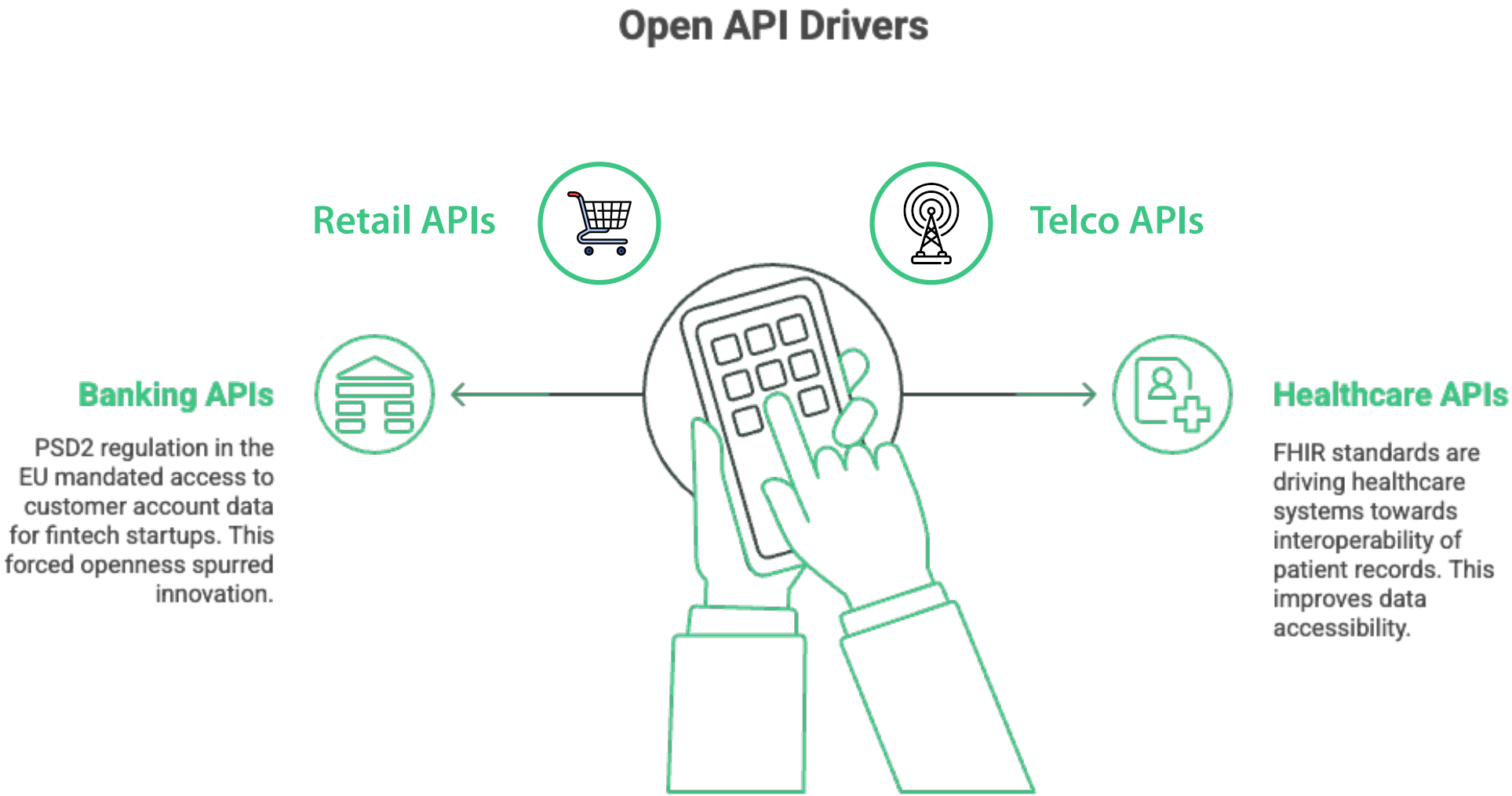
2025

Une API, c'est utile ?



Growth of API Marketplace Size

Une API, c'est utile ?



Quels sont les styles d'API ?

API: moyen de communication entre plusieurs applications

Distinction entre API et technologies

HTTP, gRPC, Kafka, REST, GraphQL sont des technologies

Quels sont les styles d'API ? (cont.)

API: c'est un langage pour interagir avec l'application !

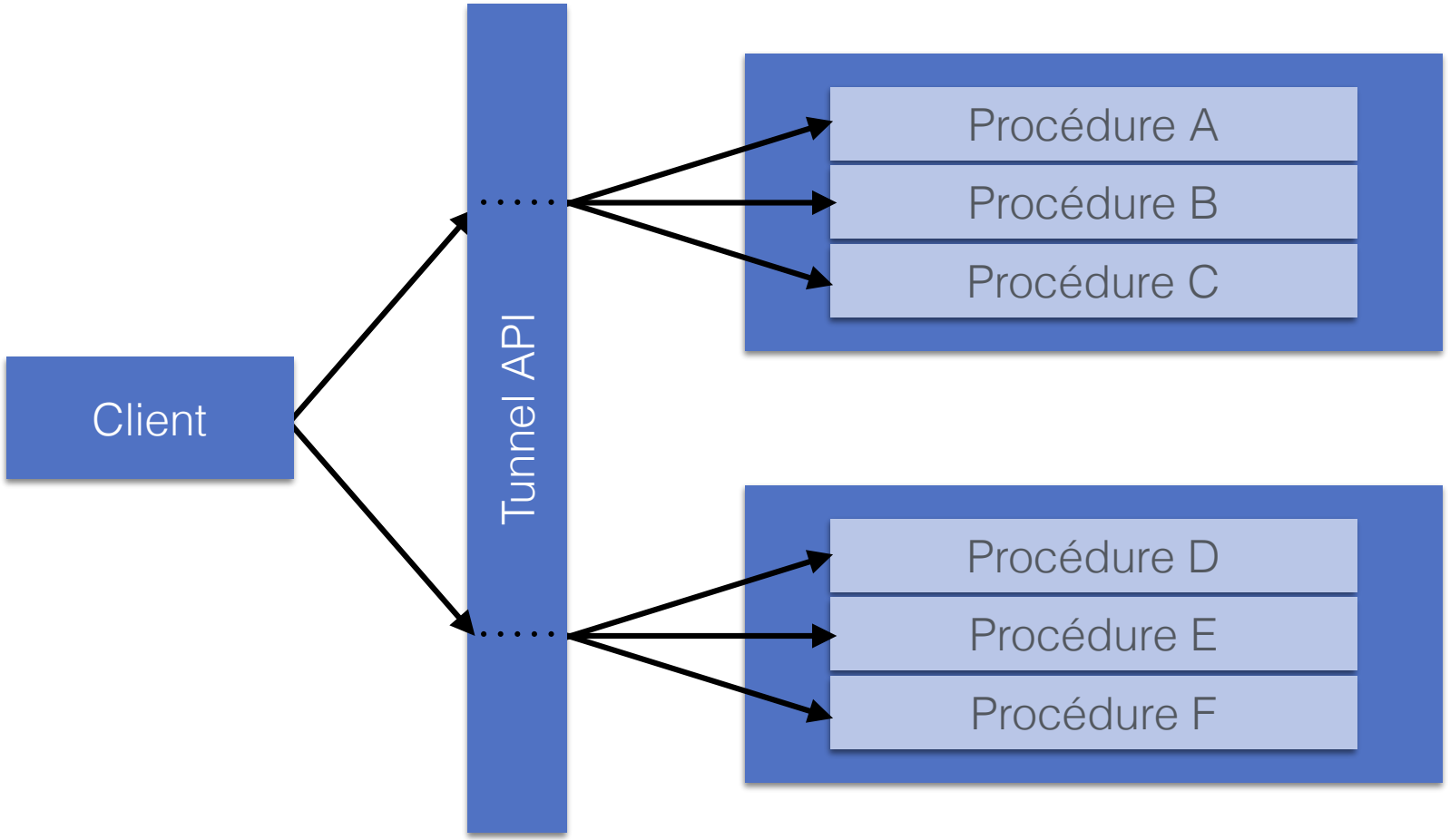
Concevoir une API équivaut à concevoir un langage

> qui sont les consommateurs ?
>
> qui sont les producteurs ?
>
> mais surtout, quel est le problème à résoudre ?

Style tunnel

Idée: appel de procédures (par ex. Google gRPC)

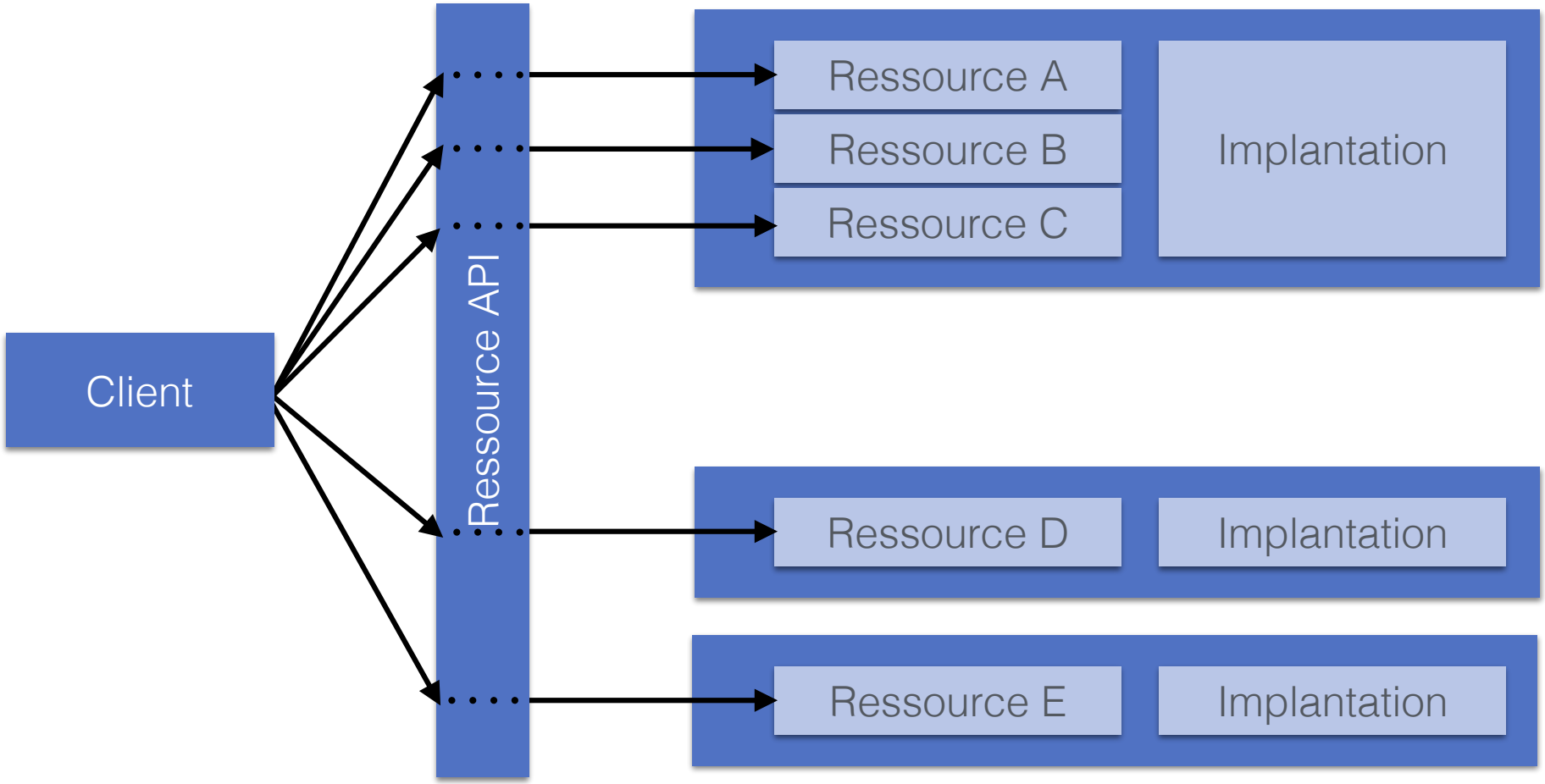
Objectif: pouvoir appeler des fonctions disponibles



Style ressource

Idée: exposer des ressources, et pouvoir interagir avec celles-ci

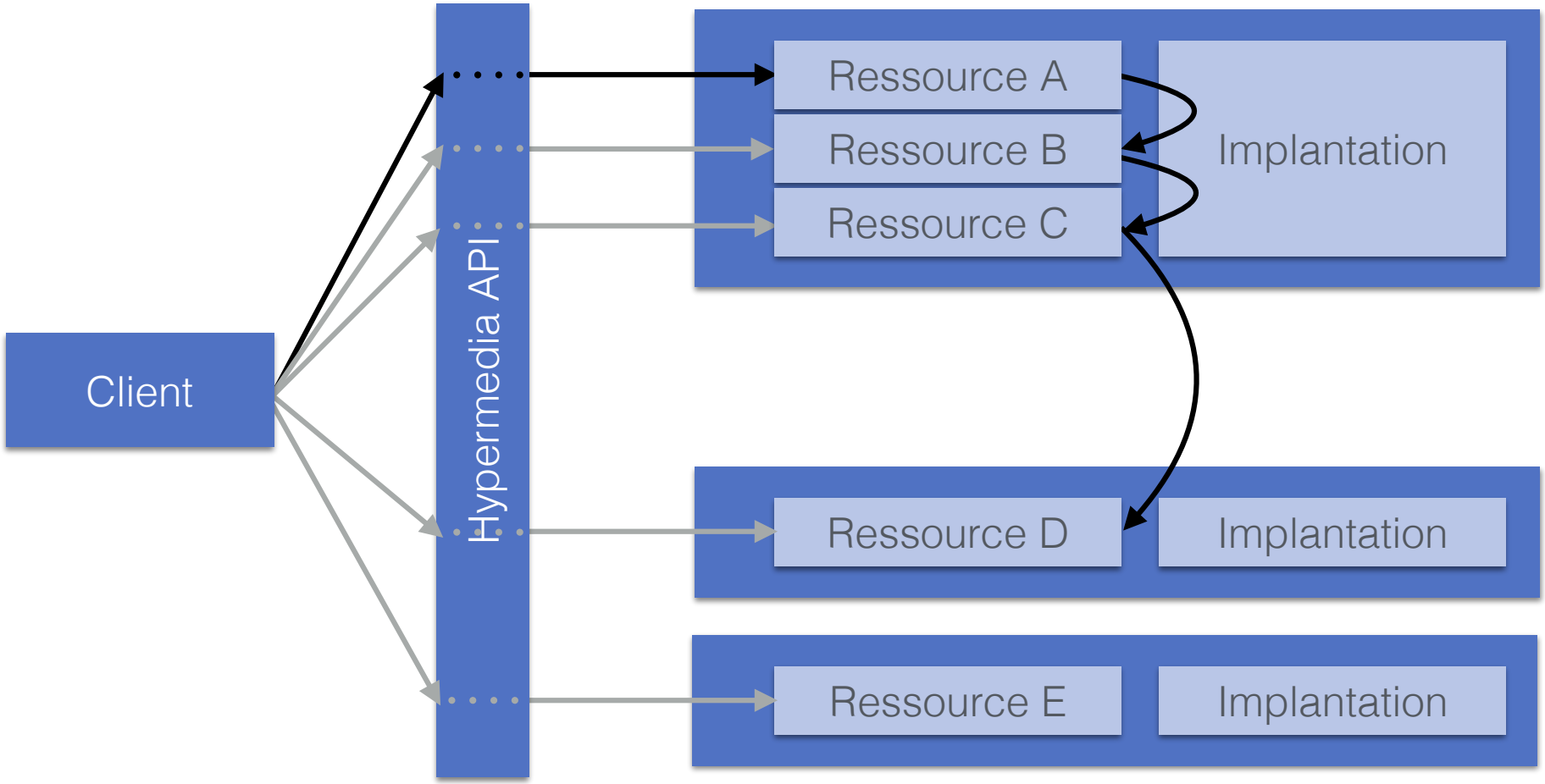
Objectif: interagir avec des ressources



Style hypermedia

Idée: basé sur les ressources, et relier les ressources entre elles

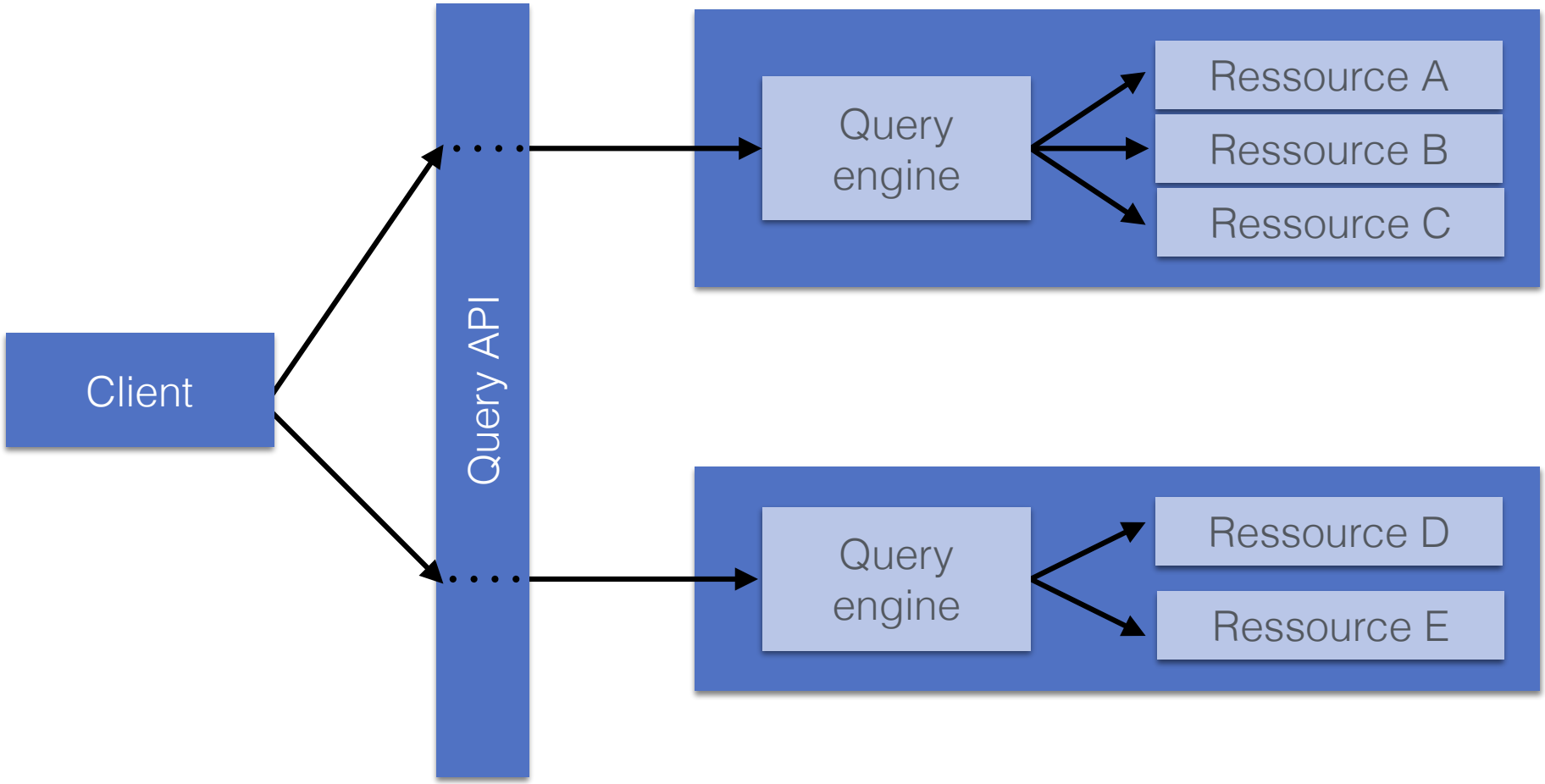
Objectif: pouvoir supporter des workflows (séquence d'opérations)



Style query

Idée: beaucoup de données, modèle de données disponible, requêtes

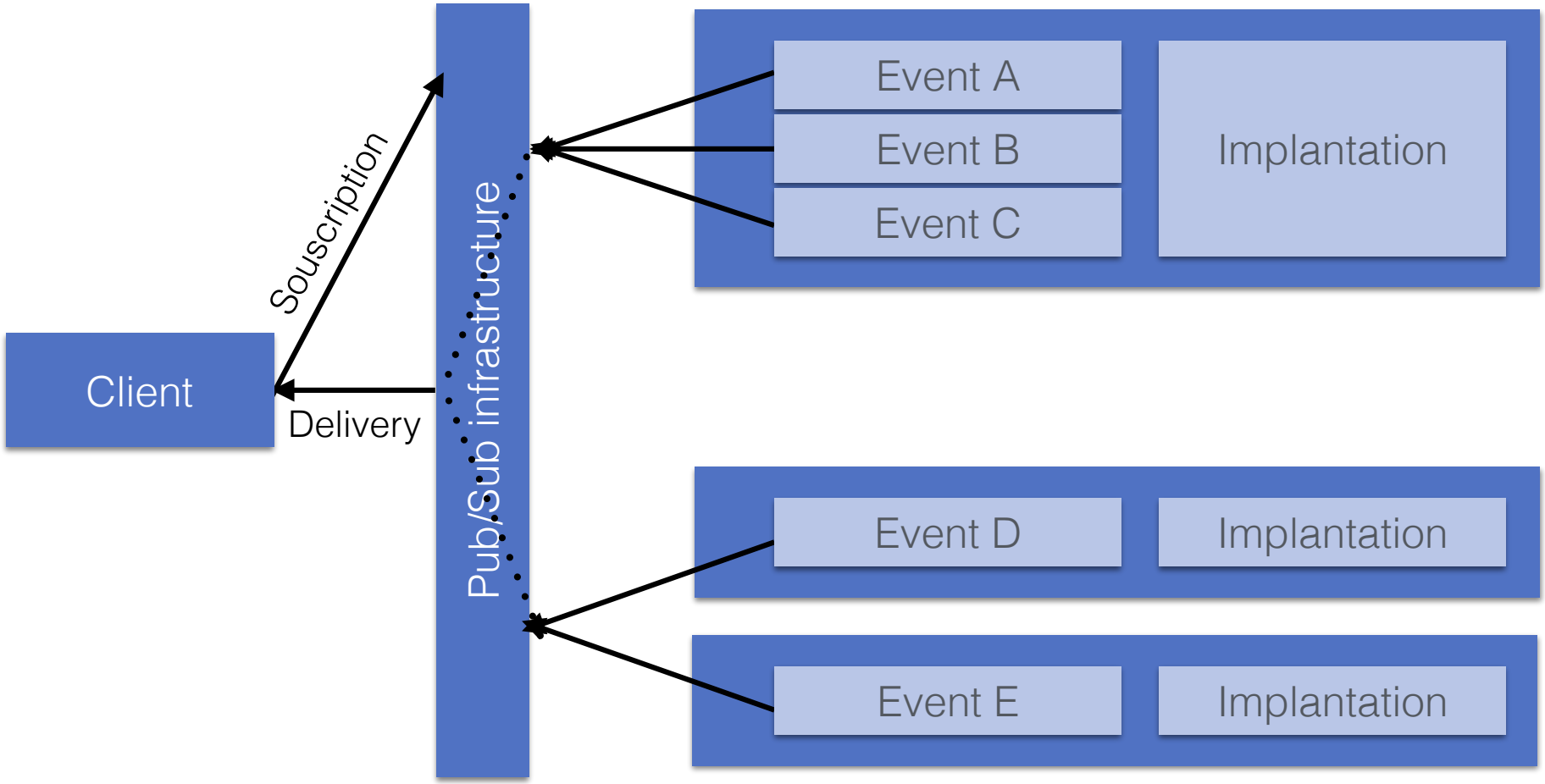
Objectif: récupérer de manière flexible et efficace des données (SQL-like)



Style event-based

Idée: inverser la communication

Objectif: pouvoir gérer beaucoup d'événements



Différents styles

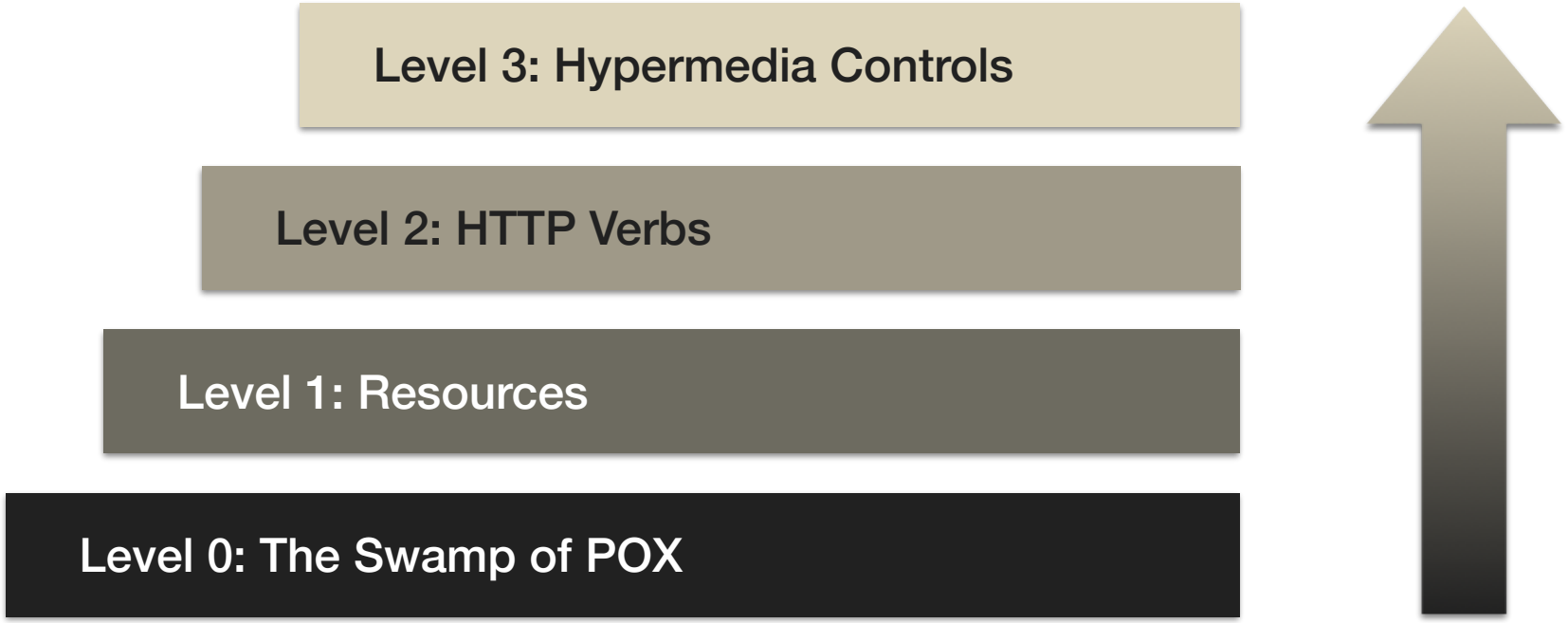
Pas de bon, pas de mauvais

Quelles sont **vos contraintes**, et quel est **votre besoin** ?

Réfléchir au **niveau technique** et au **niveau métier**

REST
REpresentational **S**tate **T**ransfer

Style d'architecture: **ressources** (URI), **représentations** (JSON, XML…), **transport**



[L. Richardson]

Quelques (**très bons**) exemples

https://developer.paypal.com/docs/api/orders/v2

https://docs.stripe.com/api/

https://docs.aws.amazon.com/apigateway/api-reference/

https://opensource.zalando.com/restful-api-guidelines/

Simplicité – **Cohérence** – **Gestion des erreurs** – **Versions**

Contraintes sur l'achitecture

client-serveur

stateless (côté serveur)

cacheable

système en couche

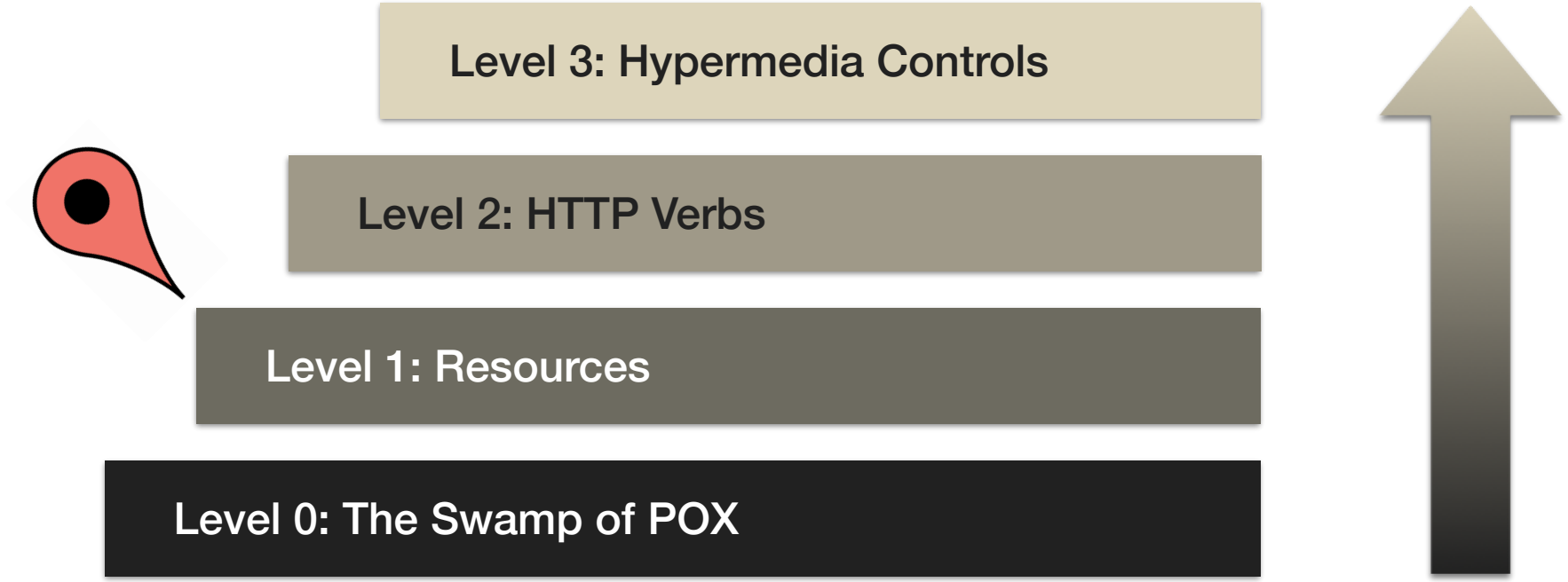
interface uniforme

Pourquoi HTTP ?

Client-serveur, uniforme, ubiquitaire, performant

HTTP: méthodes, codes d'états, headers, cache

REST: les ressources



REST: les ressources

1 ressource = 1 URI

puissance des URI pour identifier une ressource

uniformité

pas d'état dans l'adresse

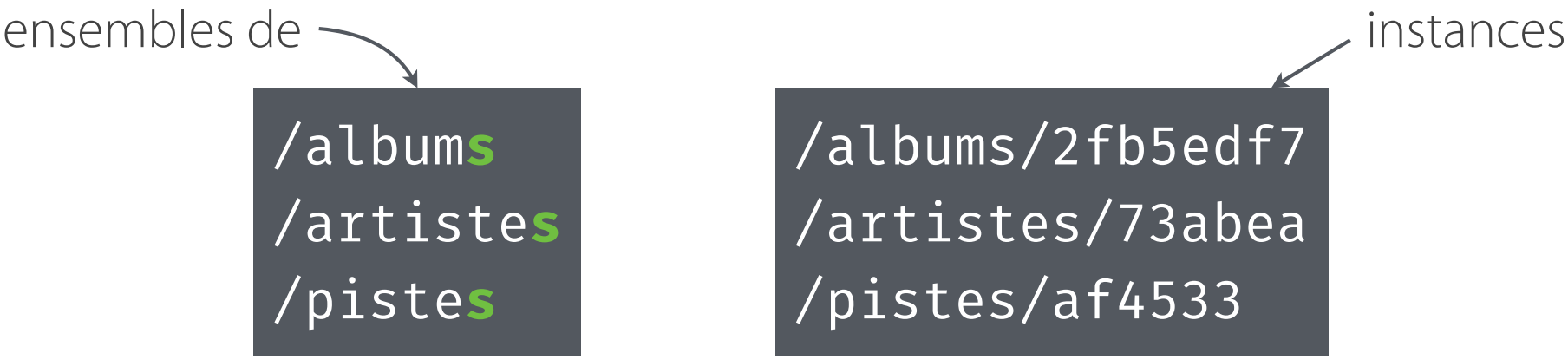
facile à gérer

information dans le message (*media type*)

REST: les ressources

Deux types de ressources

Les **collections**, et les **entités**



Singulier, pluriel ?

On utilise les formes plurielles pour les collections

```
tickets/234    vs    ticket/234
```

facilite le routage (préfixe identique)

234 pointe vers un élément de la collection de tickets

Attention aux formes particulières (en anglais)

/person vs /people

UpperCamelCase ou lowerCamelCase

snake_case ou spinal-case

Préférez les minuscules

Préférez spinal-case ou snake_case

_ est le choix de beaucoup d'APIs

Choisissez une casse et **restez cohérent** !

Les associations

/tickets/123/messages/4

un ticket peut être un groupe de messages

/usergroups/234/users/56

un utilisateur peut appartenir à plusieurs groupes

un utilisateur doit avoir sa propre URL, référencée dans le payload usergroups (ici, users/56)

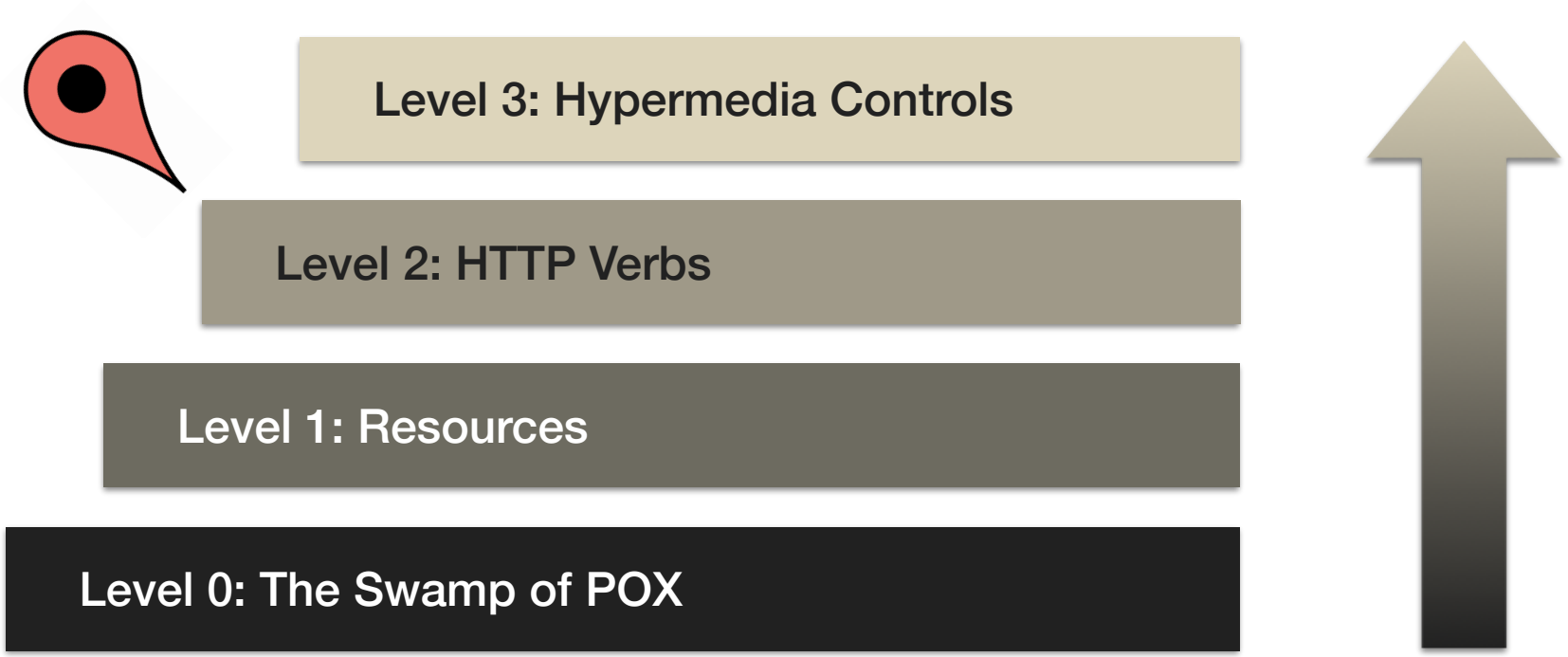
Les paramètres

Path	obligatoire, identificateur
Query	optionnel, filtre les collections
Body	logique spécifique à la ressource
Header	global, niveau plateforme

```
GET /api/tests?start=3&size=2
Host: api.example.org
HTTP/1.1 206 Partial Content

[
   "test4"
   "test5"
]
```

REST niveau 2: les actions



Pas de verbes pour les actions CRUD !

Les **noms** référencent les **ressources**

Les **ressources** sont gérées (CRUD) avec les **méthodes** HTTP

GET

POST

PUT

PATCH

DELETE

HEAD

OPTIONS

TRACE

CONNECT

GET

Récupère une collection, ou une instance de la collection

Représentation ≠ instance persistante

Mise en cache → améliore la performance

```
GET /albums
GET /albums/af54bf321
GET /albums/af54bf321/pistes/1
GET /albums/af54bf321/artistes
```

DELETE

Idempotent

Pour supprimer une ressource (suppression logique)

```
DELETE /playlists
DELETE /playlists/af3e4719ab3/pistes/3
```

PUT

On connaît l'instance (pas de mises à jour de collections)

La ressource entière doit être transmise (idempotence)

Pas de mises à jour partielles

```
PUT /playlists/af3e4719ab3
PUT /playlists
```

POST

Ajoute une instance à une **collection**, ou déclenche une action

Mais si je rappelle, je rajoute, et si je rappelle, je rajoute,…

pas idempotent

```
POST /playlists
POST /playlists/af3e4719ab3
```


PATCH

Idempotent

Pour mettre à jour partiellement des données

```
PATCH /playlists/af3e4719ab3
PATCH /playlists
```

OPTIONS

Retourne les méthodes supportées pour une URI

```
OPTIONS /api/members/42 HTTP/1.1
Host: api.example.org
HTTP/1.1 204 No Content
Allow: HEAD, GET, PUT, DELETE, OPTIONS
```

HEAD

Retourne un entête de message sans corps contenant certaines des informations obtenues avec un GET

```
GET /api/tests?start=3&size=4
Host: api.example.org
HTTP/1.1 206 Partial Content

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

```
HEAD /api/tests?start=3&size=4
Host: api.example.org
HTTP/1.1 200 OK
Accept-Ranges: data
Content-Range: data 3-7/18
```

Exercice

Pourquoi les URIs suivants posent problème ?

```
/pistes/af3e4719ab3/getDuree
/playlists/create
/playlists/af3e4719ab3/addPiste
/albums/updateReleaseDate?id_album=42
```

⚠️ Attention, un verbe peut être utilisé pour une action métier

Exemples

- /login, /logout
- /convertTemperature
- repositories/123/like

Paypal

- payments/authorizations/{authorization_id}/reauthorize

Stripe

- payment_intents/{payment_id}/cancel

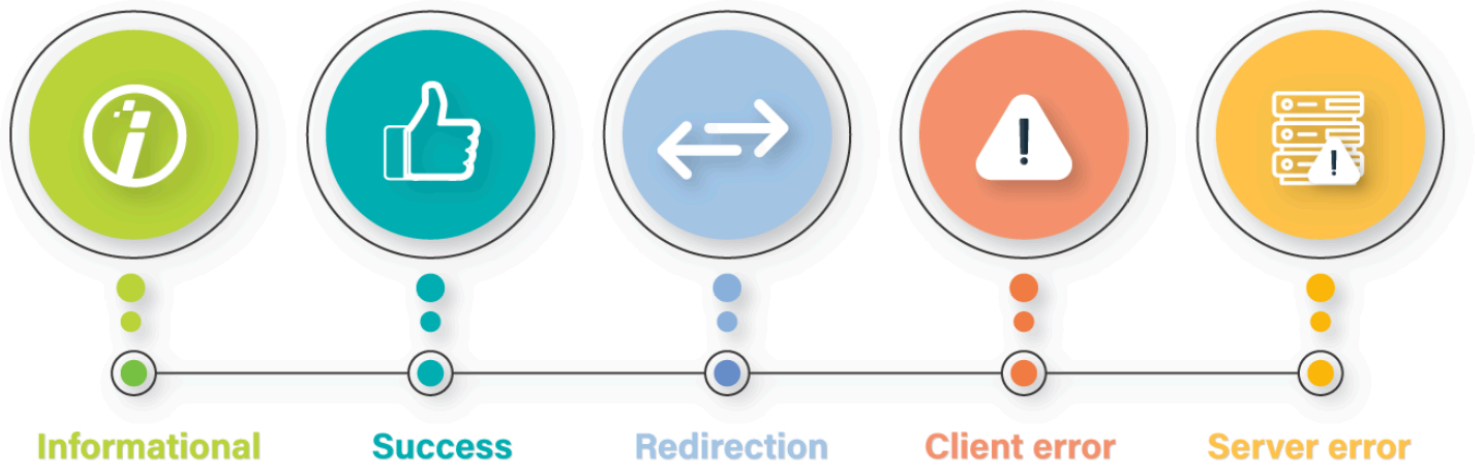
Les codes HTTP

www.iana.org/assignments/http-status-codes/http-status-codes.xhtml

geekflare.com/http-status-code-infographics/

www.loggly.com/blog/http-status-code-diagram/

webconcepts.info/concepts/http-status-code/



Quelques codes HTTP

`GET /orders/42`	`HTTP/1.1` **`200: Ok`**
`POST /playlists`	`HTTP/1.1` **`201: Created`**
`POST /orders/42/validate`	`HTTP/1.1` **`202: Accepted`**
`DELETE /orders/42`	`HTTP/1.1` **`204: No content`**
`GET /orders?offset=10`	`HTTP/1.1` **`206: Partial content`**
`GET /orders/42`	`HTTP/1.1` **`304: Not modified`**
`DELETE /orders/43`	`HTTP/1.1` **`404: Not found`**
`POST /orders/42`	`HTTP/1.1` **`405: Not allowed`**

Les erreurs (RFC 9457)

Indiquer le status et le Content-type

Associer un objet json décrivant le détail du problème à destination des utilisateurs

```
HTTP/1.1 404 Not Found
Content-Type: application/json;charset=utf-8

{
 "type": "not found error",
 "status" : 404,
 "title": "ressource inexistante"
 "detail" : "la ressource demandée n'existe pas : /albums/43"
}
```

Traitement des cas inconnus

Se ramener au cas générique

4xx -> 400

5xx -> 500

Une bonne API est adaptable

Des réponses différentes… pour des clients différents

Accept: application/json, application/xml...

Accept-Charset: utf-8, iso-8859-1

Accept-Encoding: gzip

Accept-Language: en-US, en, fr

Le tout dans le header, avec les media-types !

Versions

Le plus courant, dans l'URL:

```
https: //api.com/v2/restaurants/1234
```

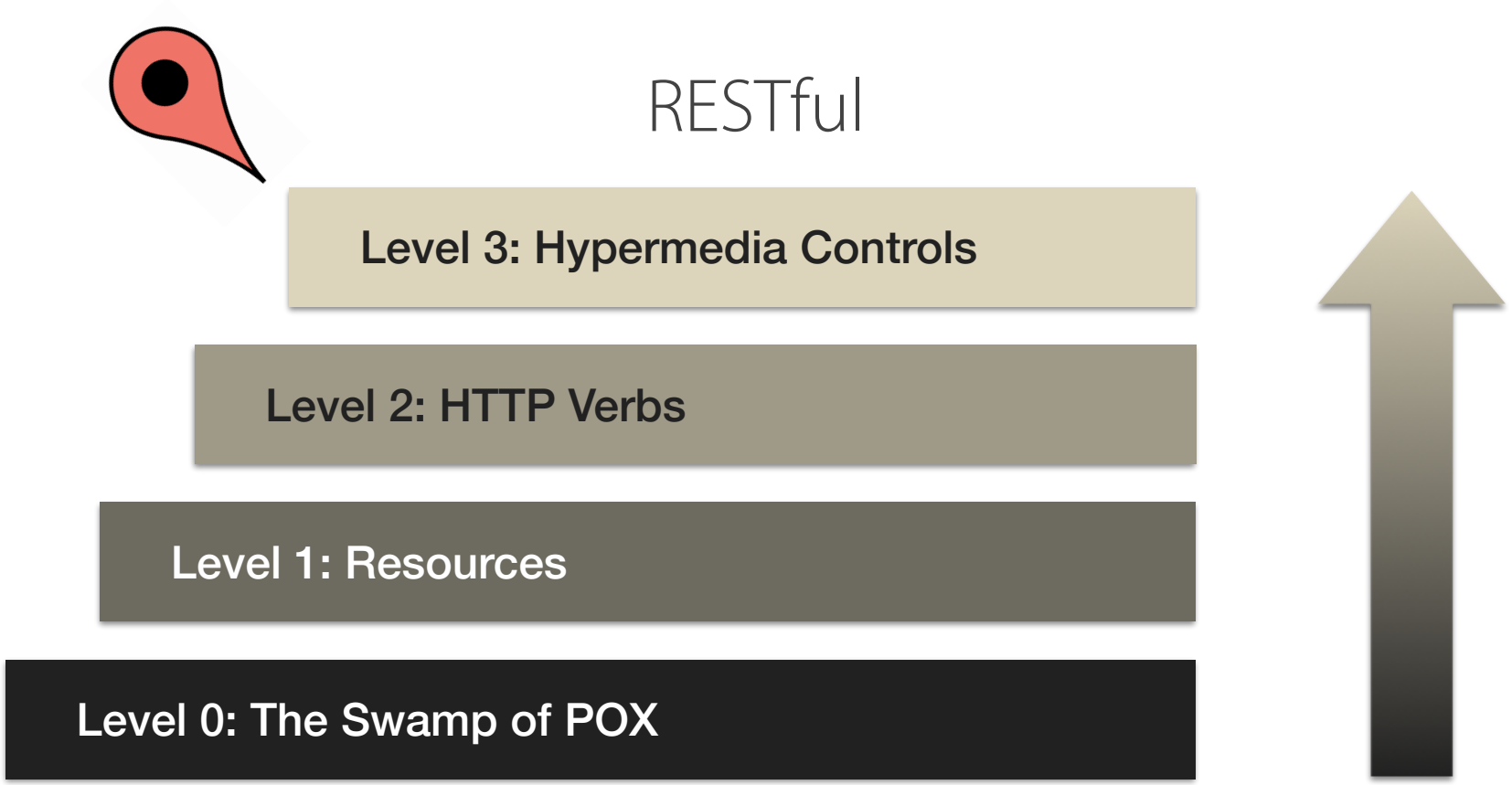
Via un attribut dans le header:

```
X-API-Version: 2
```

Avec l'attribut Accept du header:

```
GET /repos
Accept: application/vnd.github.v3+json
```

REST Niveau 3



HATEOAS

Je ne dois pas avoir à deviner…

…comment accéder à une ressource donnée

…quelles sont les relations entre une ressource et les autres ressources

API auto-descriptive !

HATEOAS (cont.)

Je modélise mon **domaine métier**

l'API ne doit pas exposer aveuglément le modèle de données mais le métier

utiliser des données hiérarchiques pour éviter les répétitions et pour clarifier le contexte

l'API doit indiquer les relations entre les ressources

établir des relations parents-enfants claires, établir des liens clairs avec d'autres ressources

l'API doit permettre de connaître le cycle de vie de mes ressources

On passe d'un REST structurel à un **REST comportemental**

HATEOAS (cont.)

Les liens servent à décrire les étapes valides

En suivant les liens, et en interagissant avec les ressources, on change l'état de l'application

Les *media types* définissent les contrats

Les liens décrivent les transitions entre les états

Il n'est pas nécessaire d'avoir une description statique du contrat (il peut évoluer avec la version)

C'est cela HATEOAS !

Hypermedia

Où est la difficulté ?

Avoir des clients génériques

Faire attention au volume de données échangées entre client et service (mobile/connectivité)

Compréhension des liens

Workflow[Webber]

Faire une sélection

    ajouter éventuellement des options

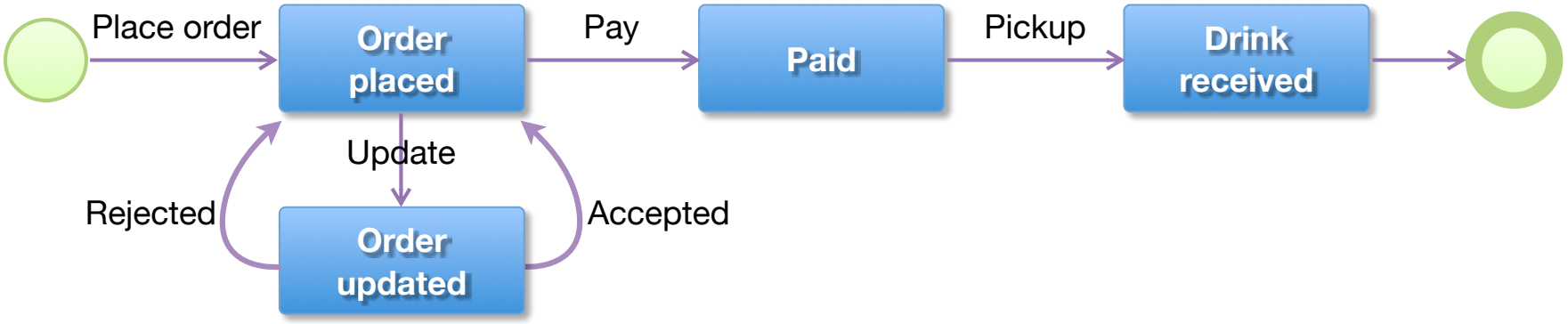
Payer

Attendre

Récupérer la boisson

Côté client

Processus: vue côté client



Story 1:

As a customer, I want to order a coffee

so that Starbucks can prepare my drink

Créer une commande

```
POST http://restbucks.com/orders HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: restbucks.com
Connection: keep-alive
Content-Length: 283

{ "lineItems" : [ {
    "name" : "Latte",
    "quantity" : 1,
    "milk" : "Semi",
    "size" : "Large",
    "price" : "EUR 4.20"
  } ],
  "location" : "Take away"
}
```

Réponse

```
HTTP/1.1 201 Created
Location: http://restbucks.com/orders/16
Content-Type: application/json
Content-Length: 1039
Date: Tue, 22 Jan 2019 08:05:49 GMT

{"order" : {
    "lineItems" : [ {
      "name" : "Latte",
      "quantity" : 1,
      "milk" : "Semi",
      "size" : "Large",
      "price" : "EUR 4.20"
    } ],
    "location" : "Takeaway",
    "orderedDate" : "2019-01-22T08:05:49.581",
    "status" : "Unpaid",
    "price" : "EUR 4.20"
  },
  "_links" : {
    "self" : {
      "href" : "http://restbucks.com/orders/16"
    },
    "restbucks:cancel" : {
      "href" : "http://restbucks.com/orders/16",
      "rel" : "http://relations.restbucks.com/cancel"
      "title" : "Cancel the order"
    },
    "restbucks:payment" : {
      "href" : "http://restbucks.com/payments/16",
      "rel" : "http://relations.restbucks.com/payment"
      "title" : "Pay the order"
    },
    "restbucks:update" : {
      "href" : "http://restbucks.com/orders/16",
      "rel" : "http://relations.restbucks.com/update"
      "title" : "Update the order"
    },
  }
}
```

Story 2:

As a customer, I want to be able to change my drink to suit my tastes

Modifier une commande

```
POST http://restbucks.com/orders/16 HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: restbucks.com
Connection: keep-alive
Content-Length: 288

{ "lineItems" : [ {
    "name" : "Cappuccino",
    "quantity" : 1,
    "milk" : "Semi",
    "size" : "Large",
    "price" : "EUR 4.20"
  } ],
  "location" : "Take away"
}
```

Réponse

```
HTTP/1.1 200 OK
Location: http://restbucks.com/orders/16
Content-Type: application/json
Content-Length: 1039
Date: Tue, 22 Jan 2019 08:07:49 GMT

{"order" : {
    "lineItems" : [ {
      "name" : "Cappuccino",
      "quantity" : 1,
      "milk" : "Semi",
      "size" : "Large",
      "price" : "EUR 4.20"
    } ],
    "location" : "Takeaway",
    "orderedDate" : "2019-01-22T08:05:49.581",
    "status" : "Unpaid",
    "price" : "EUR 4.20"
  },
  "_links" : {
    "self" : {
      "href" : "http://restbucks.com/orders/16"
    },
    "restbucks:cancel" : {
      "href" : "http://restbucks.com/orders/16",
      "rel" : "http://relations.restbucks.com/cancel"
      "title" : "Cancel the order"
    },
    "restbucks:payment" : {
      "href" : "http://restbucks.com/payments/16",
      "rel" : "http://relations.restbucks.com/payment"
      "title" : "Pay the order"
    },
    "restbucks:update" : {
      "href" : "http://restbucks.com/orders/16",
      "rel" : "http://relations.restbucks.com/update"
      "title" : "Update the order"
    },
  }
}
```

POST ou PUT ?

Safe - si une méthode est invoquée, l'état de la ressource reste inchangé

Idempotent - peu importe le nombre d'invocation de la méthode, le résultat reste le même

POST ou PUT ?

POST n'est pas idempotent, PUT l'est !

PUT a besoin de la ressource complète

Mais ici, le client ne connait pas le prix ou l'état de la commande

PATCH serait idéal, car il permet de n'envoyer que ce qui change

Stateless

Aie !

Stateless

Interactions sans état ! Les ressources « vous oublient »

Du coup, possibilité de *race conditions*

On utilise :

If-Unmodified-Since avec une estampille

ou **If-Match** et un **ETag**

Vous recevez un 412 PreconditionFailed si vous avez perdu !

Mais vous êtes sûr d'avoir la ressource dans un état cohérent

Supprimer une commande

```
DELETE http://restbucks.com/orders/16 HTTP/1.1
Host: restbucks.com
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Date: Tue, 22 Jan 2019 08:09:22 GMT
```

```
HTTP/1.1 200 OK
Location: http://restbucks.com/orders/16
Content-Type: application/json
Content-Length: 1039
Date: Tue, 22 Jan 2019 08:07:49 GMT

{"order" : {
    "lineItems" : [ {
      "name" : "Cappuccino",
      "quantity" : 1,
      "milk" : "Semi",
      "size" : "Large",
      "price" : "EUR 4.20"
    } ],
    "location" : "Takeaway",
    "orderedDate" : "2019-01-22T08:05:49.581",
    "status" : "Unpaid",
    "price" : "EUR 4.20"
  },
  "_links" : {
    "self" : {
      "href" : "http://restbucks.com/orders/16"
    }
    // ...
    "restbucks:payment" : {
      "href" : "http://restbucks.com/payments/16",
      "rel" : "http://relations.restbucks.com/payment"
      "title" : "Pay the order"
    },
    // ...
  }
}
```

Ok, je veux payer la commande

Story 3:

As a customer, I want to be able to pay my bill to receive my drink

Quelle méthode ?

```
PUT http://restbucks.com/payments/16 HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: restbucks.com
Connection: keep-alive
Content-Length: 287

{"payment": {
  "amount": "EUR 4.20",
  "cardHolderName": "Michael Farraday",
  "creditCardNumber": "1234123412341234",
  "expiryMonth": "December",
  "expiryYear": "2020"
  }
}
```

Réponse

```
HTTP/1.1 200 OK
Location: http://restbucks.com/payments/16
Content-Type: application/json
Content-Length: 650
Date: Tue, 22 Jan 2019 08:58:03 GMT

{"payment": {
   "amount": "EUR 4.20",
   "cardHolderName": "Michael Farraday",
   "creditCardNumber": "1234123412341234",
   "expiryMonth": "December",
   "expiryYear": "2020"
   },
     "_links" : {
       "self" : {
         "href" : "http://restbucks.com/payments/16"
       }
       "restbucks:receipt" : {
         "href" : "http://restbucks.com/receipts/16",
         "rel" : "http://relations.restbucks.com/receipt"
         "title" : "Get the receipt"
       },
       "restbucks:order" : {
         "href" : "http://restbucks.com/orders/16",
         "rel" : "http://relations.restbucks.com/order"
         "title" : "Get the order"
       }
}
```

Et si je veux annuler maintenant ?

```
DELETE http://restbucks.com/orders/16 HTTP/1.1
Host: restbucks.com
```

```
HTTP/1.1 405 Method Not Allowed
Allow: GET
Content-Type: application/json
Content-Length: 0
Date: Tue, 22 Jan 2019 08:59:29 GMT
```

Récupérer ma facture

```
GET http://restbucks.com/receipts/16 HTTP/1.1
Accept: application/json
Host: restbucks.com
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Date: Tue, 22 Jan 2019 09:00:03 GMT

{"receipt": {
  "amount": "EUR 4.20",
  "paid": "2019-01-22T08:58:54.651",
    },
    "_links" : {
      "self" : {
        "href" : "http://restbucks.com/receipts/16"
      }
      "restbucks:order" : {
        "href" : "http://restbucks.com/orders/16",
        "rel" : "http://relations.restbucks.com/order"
        "title" : "Get the order"
      }
   }
}
```

Ma commande est-elle prête ?

```
GET http://restbucks.com/orders/16 HTTP/1.1
Host: restbucks.com
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 534
Date: Tue, 22 Jan 2019 09:05:12 GMT

{"order" : {
    "lineItems" : [ {
      "name" : "Cappuccino",
      "quantity" : 1,
      "milk" : "Semi",
      "size" : "Large",
      "price" : "EUR 4.20"
    } ],
    "location" : "Takeaway",
    "orderedDate" : "2019-01-22T08:05:48.694",
    "status" : "Ready",
    "price" : "EUR 4.20"
  },
  "_links" : {
    "restbucks:receipt" : {
      "href" : "http://restbucks.com/receipts/16",
      "rel" : "http://relations.restbucks.com/receipt"
      "title" : "Get the receipt"
    }
 }
```

Ok, tout est parfait !

```
GET http://restbucks.com/receipts/16 HTTP/1.1
Host: restbucks.com
```

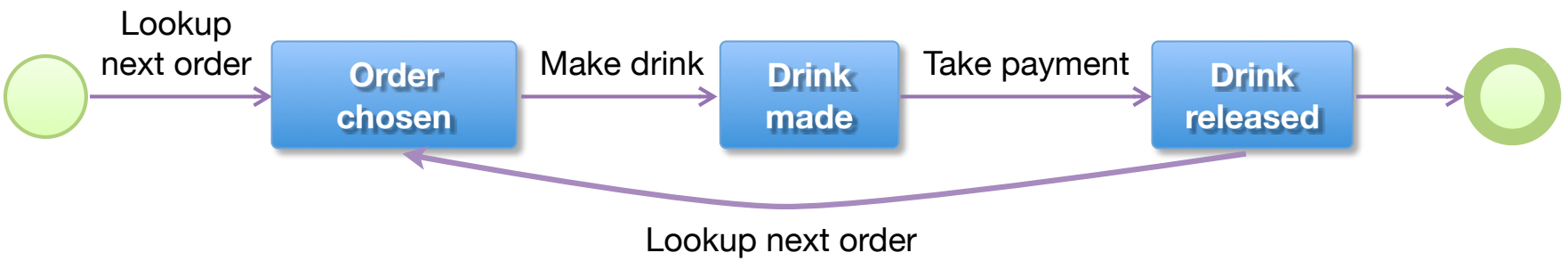
```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 393
Date: Tue, 22 Jan 2019 09:07:34 GMT

{"order" : {
    "lineItems" : [ {
      "name" : "Cappuccino",
      "quantity" : 1,
      "milk" : "Semi",
      "size" : "Large",
      "price" : "EUR 4.20"
    } ],
    "location" : "Takeaway",
    "orderedDate" : "2019-01-22T08:05:48.694",
    "status" : "Delivered",
    "price" : "EUR 4.20"
  }
}
```


ARBUCKS
OFFEE

Et côté Starbucks ?

Processus: vue côté Starbucks



Story 4:

As a barista, I want to see the list of drinks that I need to make, so that I can serve my customers

Et côté Starbucks ?

```
GET /orders HTTP 1.1
Host: internal.restbucks.com
```

Et côté Starbucks ?


```
HTTP/1.1 200 OK
Location: http://internal.restbucks.com/orders/16
Content-Type: application/json
Date: Tue, 22 Jan 2019 08:05:49 GMT

{"order" : {
    "orderedDate" : "2019-01-22T08:05:49.581",
    "status" : "Placed",
 },
  "_links" : {
    "self" : {
      "href" : "http://internal.restbucks.com/orders/16"
    },
    "restbucks:details" : {
      "href" : "http://internal.restbucks.com/orders/16",
      "rel" : "http://relations.restbucks.com/details"
      "title" : "Details of the order"
    }
  }
}
```

Et les détails ?

```
GET /orders/16/details HTTP/1.1
Host: internal.restbucks.com
```

Et les détails ?

```
HTTP/1.1 200 OK
Date: ...
Content-Length: ...
Content-Type: application/json

{"order" : {
    "lineItems" : [ {
      "name" : "Cappuccino",
      "quantity" : 1,
      "milk" : "Semi",
      "size" : "Large",
      "price" : "EUR 4.20"
    },
    {
      "name" : "Latte",
      "quantity" : 2,
      "milk" : "Semi",
      "size" : "Small",
      "price" : "EUR 5.80"
    } ],
    "_links" : {
      "self" : {
        "href" : "http://internal.restbucks.com/orders/16"
      },
      "restbucks:fulfill" : {
        "href" : "http://internal.restbucks.com/orders/16",
        "rel" : "http://relations.restbucks.com/fulfillment"
        "title" : "Fulfill the order"
      }
    }
}
```

Story 5:

As a barista, I want to check that a customer has paid for their drink so that I can serve it

Et le paiement ?

```
GET /payments/16 HTTP 1.1
Host: internal.restbucks.com
Authorization: Digest username="serveur joe"
realm="internal.restbucks.com"
nonce="..."
...
```

Story 6:

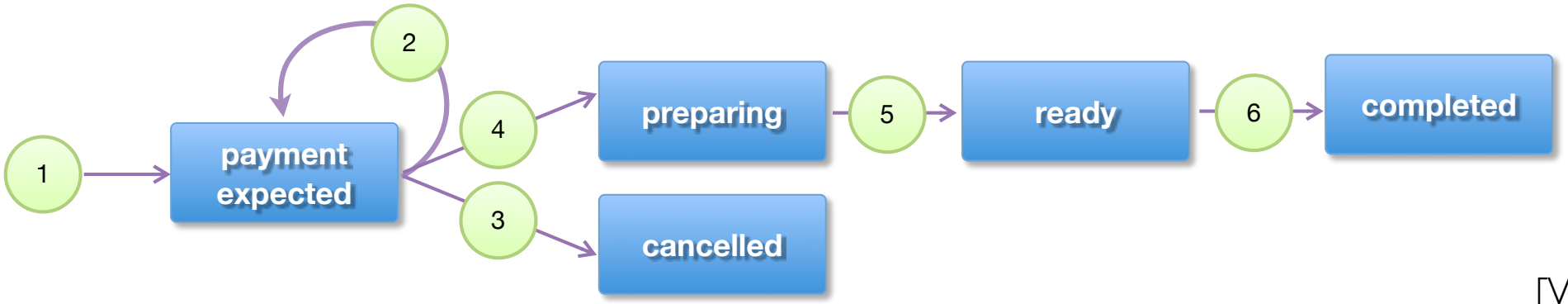
As a barista, I want to remove drinks I have made from the pending list so that I don't make duplicates

C'est bon, au suivant…

```
DELETE /orders/16 HTTP 1.1
Host: internal.restbucks.com
```

POST	http://restbucks/order	new order (create resource)	1
POST	http://restbucks/order/1234	update order (only if state is at « payment expected »	2
DELETE	http://restbucks/order/1234	order cancelled (only if state is at « payment expected) »	3
PUT	http://restbucks/payment/1234	Payment accepted	4
		Order preparation	5
DELETE	http://restbucks/receipt/1234	Conclude the ordering process	6
GET	http://restbucks/order/1234	Return the latest representation of the resource	

Logique métier



[Webber]