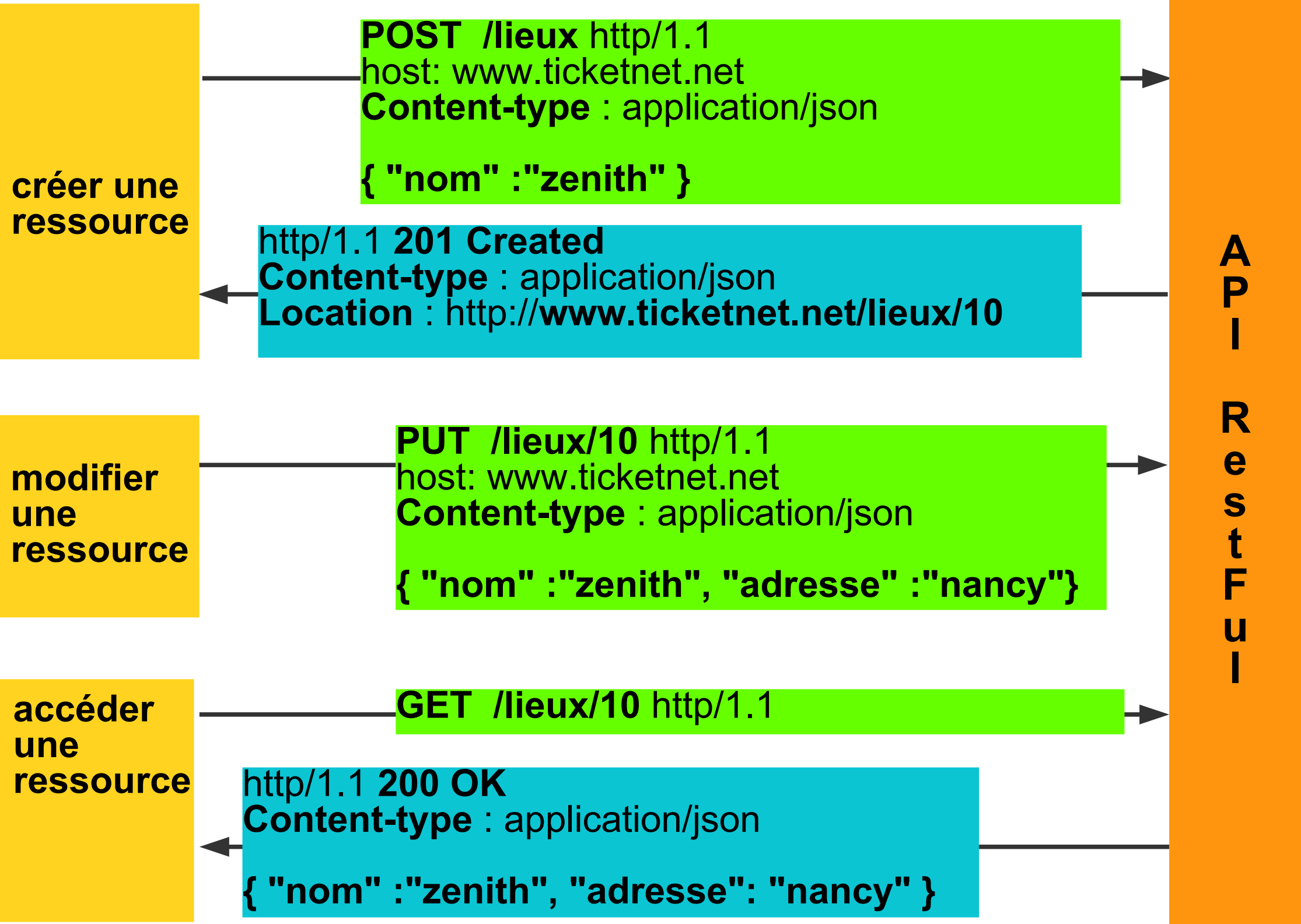
Construire des api REST

Représentation des ressources, personnalisation des réponses

le modèle REST

le client et le serveur communiquent en échangeant une ***représentation*** de l'état des ressources :

- lire une ressource : **transférer** une **représentation** de son **état courant** du serveur vers le client
- créer une nouvelle ressource : **transférer** une **représentation** de son **état initial** du client vers le serveur
- remplacer une ressource : **transférer** une **représentation** de son **nouvel état** du client vers le serveur


créer une ressource
POST /lieux http/1.1
host: www.ticketnet.net
Content-type : application/json
{ "nom" :"zenith" }
http/1.1 201 Created
Content-type : application/json
Location : http://www.ticketnet.net/lieux/10
modifier une ressource
PUT /lieux/10 http/1.1
host: www.ticketnet.net
Content-type : application/json
{ "nom" :"zenith", "adresse" :"nancy"}
accéder une ressource
GET /lieux/10 http/1.1
http/1.1 200 OK
Content-type : application/json
{ "nom" :"zenith", "adresse": "nancy" }
API RestFul

représentation des ressources/collections

- Représentation de l'état d'une ressource/collection échangé dans les requêtes/réponses :
  - Etat de la ressource suite à GET
  - nouvel état d'une ressource dans un PUT
  - description de la nouvelle ressource dans un POST

- représentation dans un format neutre : XML, JSON
  - privilégier json,
  - XML : version minimaliste sans namespace, pas de DTD
  - si possible proposer les 2, choix par le client grâce au header `Accept:`

ressource (json) :

```
{
    "id": 1,
    "nom": "le forestier",
    "description": "le bon sandwich au … ",
    "img": {
            "href": "/img/le_forestier.png"
      }
}
```

ressource (xml) :

```
<sandwich>
 <id>1</id>
 <nom>Le forestier</nom>
 <description>"le bon sanwich au goût ..  "
 </description>
 <img xlink:href="/img/le_forestier.png">
</sandwich>
```

collection (json)

```
[
  {
   "sandwich" : {
      "id": 1,
      "nom" : "le forestier"
    },
   "links": {"self" : { "href":"/sandwichs/1" } }
  }
 ,
  {
   "sandwich" : {
      "id":2,
      "nom" : "le sous-marin"
     },
   "links" : {"self" :  {"href":"/sandwichs/2"}}
   }
]
```

rappel

- Le modèle des ressources exposées par l'api, et donc la représentation des ressources échangées entre les clients et les serveurs n'est pas identique au modèle de la base de données où les ressources sont stockées
  - Présenter une vue des données plus facile à utiliser pour développer les applications clientes
  - Toutes les données ne sont pas utiles à toutes les applications
  - Rendre l'api indépendante du schéma de la BD pour faciliter les évolutions
  - Cacher l'implantation physique pour la sécurité

contenu des **réponses** REST

- Une réponse REST est **toujours** un **objet json** regroupant **:**
- 1 ressource ou une collection,
- des méta-données décrivant la réponse : nombre d'éléments dans la collection, informations de localisation …
- des liens vers des ressources associées
  - "HATEOAS" : Hyperlink As The Engine Of Application State

Réponse REST retournant 1 ressource

GET /sandwichs/1 HTTP/1.1

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8

{ "type": "ressource",
  "locale" : "fr-FR",
  "sandwich": {
    "id": 1,
    "nom": "le forestier",
    "description": "le bon sandwich ...",
    "img": { "href": "/img/le_forestier.png" }
    },
   "links" : {
    "self" :{"href":"/sandwichs/1" },
    "categories" :{
        "href":"/sandwichs/1/categories"
  } }
}
```

Réponse REST retournant 1 collection

GET /sandwichs HTTP/1.1

```
{ "type": "collection",
  "count" : 42,
  "sandwichs":[
    { "sandwich" : {
        "id": 1,
        "nom" : "le forestier"
      },
      "links": {"self": {"href":"/sandwichs/1" }}
    },
    { "sandwich" : {
      "id":2,
      "nom" : "le sous-marin"
      },
      "links": {"self" :  {"href":"/sandwichs/2"}}
    } ] ,
   "links" : { ...}
}
```

Réponses dans les cas d'erreurs

- status indiquant l'erreur + Content-type +
- 1 objet json contenant un message « verbeux » dans la réponse, à destination des utilisateurs

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

{
 "type": "error",
 "error" : 404,
 "message": "la ressource demandée n'existe pas :
/sandwichs/1024"
}
```

Personnalisation des réponses

- L'api peut permettre de personnaliser la réponse à une requête GET : :
  - sélectionner les attributs attendus dans la réponse
  - filtrer / sélectionner le contenu d'une collection sur un ou plusieurs critères
  - trier une collection selon un critère
  - paginer une collection si elle est de grande taille
  - imbriquer des ressources associées

- **principe : utiliser la partie query de l'URI de base, ne pas créer d'uri spécifique**

sélection d'attributs

- **réponse partielle** : la requête indique les champs souhaités dans la réponse -

```
GET /sandwichs/42?fields=id,nom,img


HTTP/1.1 200 OK
Content-Type: application/json

{ "type": "ressource",
  "locale" : "fr-FR",

  "sandwich": {
      "id": 42,
      "nom": "le forestier",
      "img": { "href": "/img/le_forestier.png" }
      },

   "links" : { … }
}
```

filtrage de données

- condition sur une collection pour réaliser un filtrage ou une recherche
- **filtrage simple** sur valeur d'un ou plusieurs attributs :
  - ajouter 1 paramètre d'url pour chaque attribut sur lequel on filtre
  - /sandwichs/**?type_pain=baguette**

- **La réponse est une collection**

```
GET /sandwichs/?t=baguette

{ "type": "collection",
  "count" : 2,

  "sandwichs":[
    { "sandwich" : {
        "id": 2,
        "nom" : "le jambon-beurre"
      },
      "links": {"self": {"href":"/sandwichs/2" }}
    },
    { "sandwich" : {
        "id":12,
        "nom" : "le poulet-tomate"
      },
      "links": {"self" : {"href":"/sandwichs/12"}}
    } ] ,
    "links" : { ...}
}
```

tri

- utiliser 1 paramètre générique `sort` suivi des attributs de tri, éventuellement avec une marque indiquant le sens
  - `/sandwichs/?`**`sort=type_pain`**

    retourne la liste des jeux ordonnés par année descendante puis par nom

pagination

- pour les collections de **grande taille**, on **pagine** le résultat et on ajoute des paramètres pour indiquer
  - le nombre d'éléments attendus dans chaque page du résultat dans le résultat : size ou limit
  - le n° de page ou le nombre d'éléments à sauter : page, skip ou offset
  - /sandwichs/**?page=3&size=15**
- **prévoir des valeurs par défaut**

- Lorsque le résultat est paginé, on ajoute des liens de pagination dans les réponses

```
GET /sandwichs/?page=3&size=10 HTTP/1.1

{ "type" : "collection",
  "count": 67,
"sandwichs" : [
  {
   "sandwich" : {
      "id": 21,
      "nom" : "le poulet curry"
    },
   "links": {"self" : { "href":"/sandwichs/21" } }
  } ,
  {
   "sandwich" : {
      "id":22,
      "nom" : "le thon-crudités"
    },
   "links" : {"self" :  {"href":"/sandwichs/22"}}
   } , ...
 ],

"links" : {
    "prev" : { "href" : "/sandwichs/?page=2&size=10" },
    "next" : { "href" : "/sandwichs/?page=4&size=10" },
    "first": { "href" : "/sandwichs/?page=1&size=10" },
    "last" : { "href" : "/sandwichs/?page=7&size=7" }
 }
}
```

ressources imbriquées

- certaines associations sont utilisées de manière fréquentes dès que l'on accède 1 ressource, en particulier si cette ressource à peu d'intérêt seule
  - 1 sandwich → les catégories auxquelles il appartient

- pour éviter de faire plusieurs requêtes à chaque accès, on peut imbriquer les ressources associées dans la ressource initiale

```
GET /sandwichs/42
GET /sandwichs/42/categories
```

➔

```
GET /sandwichs/42?embed=categories
```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8

{ "type": "ressource",
  "locale" : "fr-FR",
  "sandwich": {
     "id": 1,
     "nom": "le forestier",
     "description": "le bon sandwich ...",
     "img": { "href": "/img/le_forestier.png" }
     "categories" : [
        { "categorie" : {
              "id": 1,
              "nom" : "vegetarien"
          },
          "links": {"self" :{"href":"/categories/2"}}
        },
        { "categorie" : {
              "id":3,
              "nom" : "bio"
          },
          "links" : {"self" : {"href":"/categories/3"}}
        }]
  },
```