Introduction aux services Web

Quentin Cabanes

Sommaire

1. Qu’est ce qu’un service Web

Une courte définition des services Web.

2. Pourquoi l’utiliser

Les arguments principaux montrant la puissance des services Web.

3. Mode d’emploi

Apprenez comment un service web fonctionne et comment l’utiliser.

4. Conception

Une courte explication sur la conception URL pour services Web.

5. Sécurité

Présentation de quelques principes de sécurité pour votre culture générale.

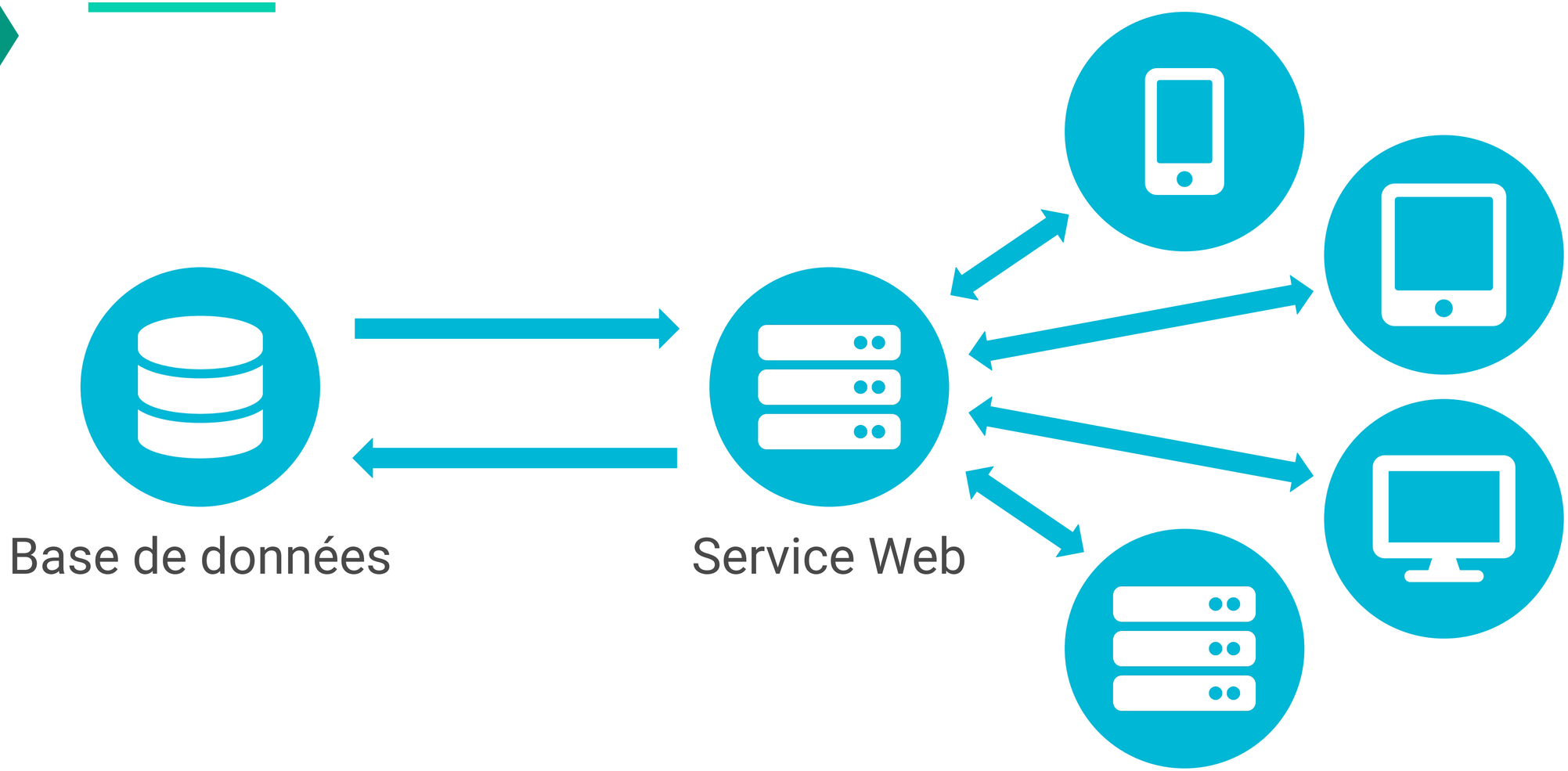





“Un service Web est un système logiciel conçu pour prendre en charge des interactions interopérables entre machines sur un réseau.”

Source: W3C

Qu’est ce qu’un service Web

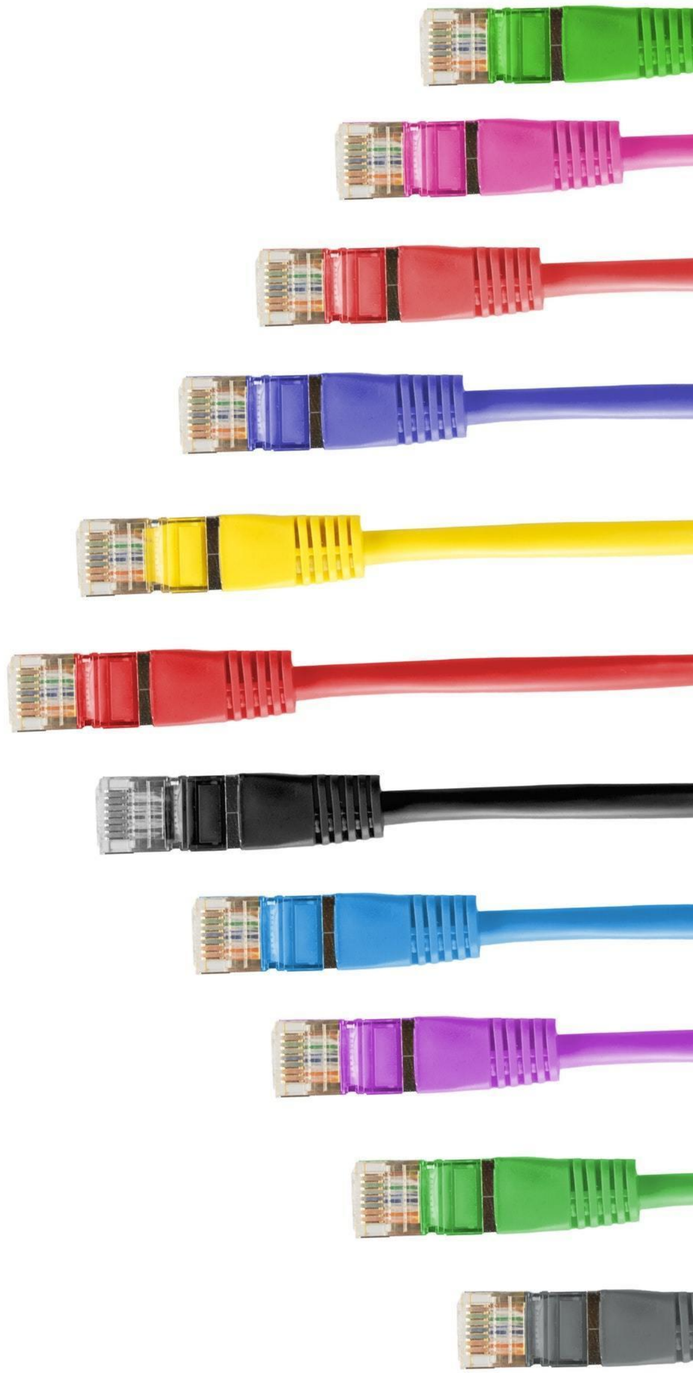


Pourquoi l’utiliser

Interopérable
N'importe quel système d'information avec un accès réseau peut être interfacé.

Modulaire
L'interface n'est pas dépendant des données et de leurs traitements.

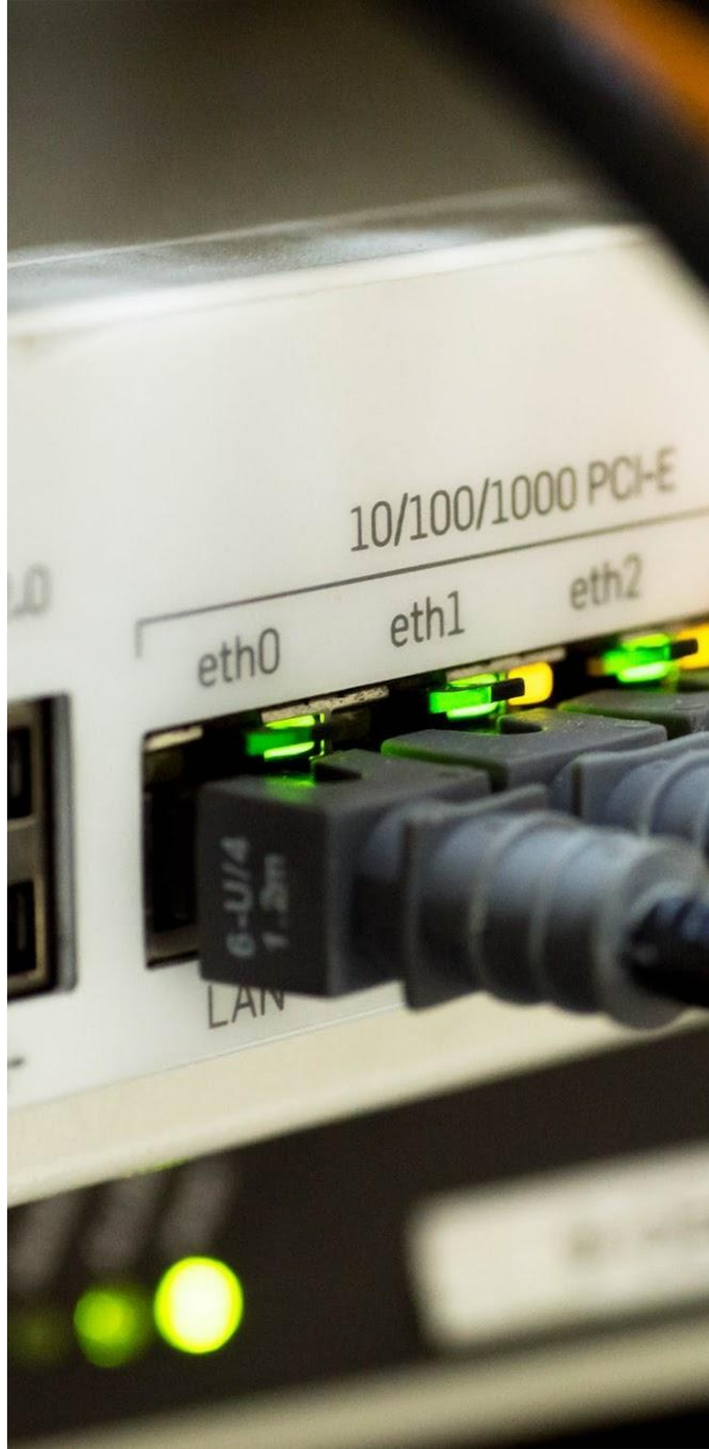
Evolutif
Les fonctionnalités peuvent changer tant que l'interfaçage n'est pas affecté.



Mode d’emploi

Representational State Transfer (REST)
Un style d’architecture qui définit des contraintes à suivre lors de la création d’un service Web.

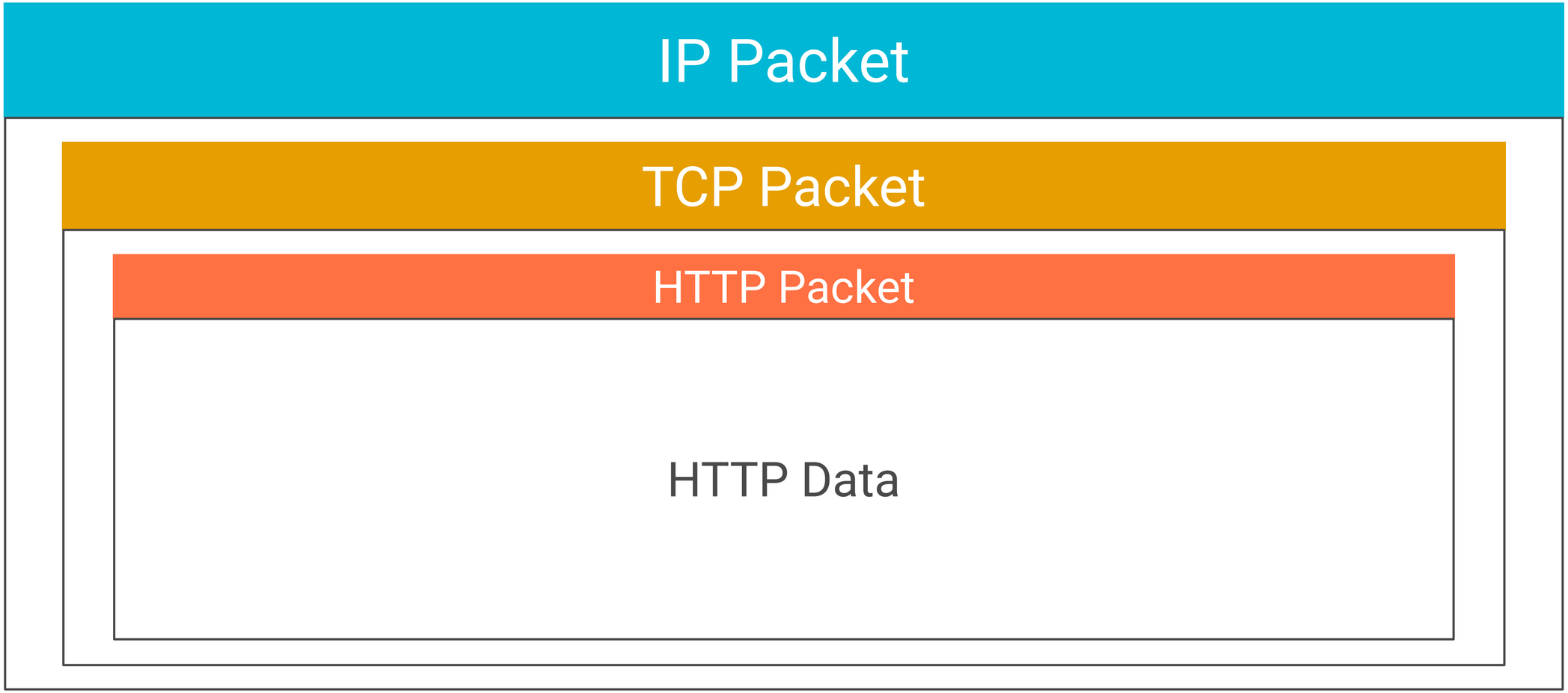
1. Le client envoie une requête HTTP à l'URL choisi
2. Le serveur reçoit la requête et la traite
3. Le serveur envoie la réponse au client
4. Le client reçoit la réponse et la traite



Mode d’emploi

Couches hôte	Application	HTTP, FTP, SSH
	Présentation	SSL, SSH, IMAP, JPEG
	Session	Sync & send to port
	Transport	TCP, UDP
Couches physique	Réseau	IP, ICMP
	Liaison	Ethernet, Switch
	Physique	Coax, Fibre optique

Mode d’emploi



Mode d’emploi

HTTP Request

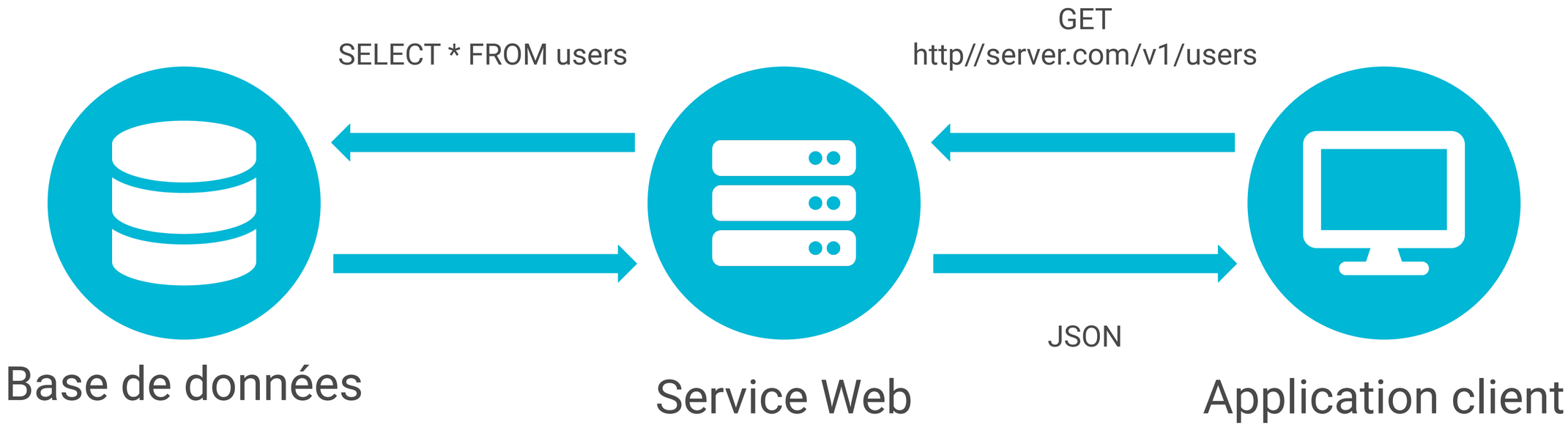
```
GET /page.php HTTP/1.1
Host: www.example.com
Connection: keep-alive
```

HTTP Response

```
HTTP/1.1 200 OK
Date: Sun, 10 Oct 2010 23:26:07
GMT
Server: Apache/2.2.8 (Ubuntu)
Content-Length: 41
Content-Type: text/html

<html>
  <body>Hello world!<body>
</html>
```

Mode d’emploi





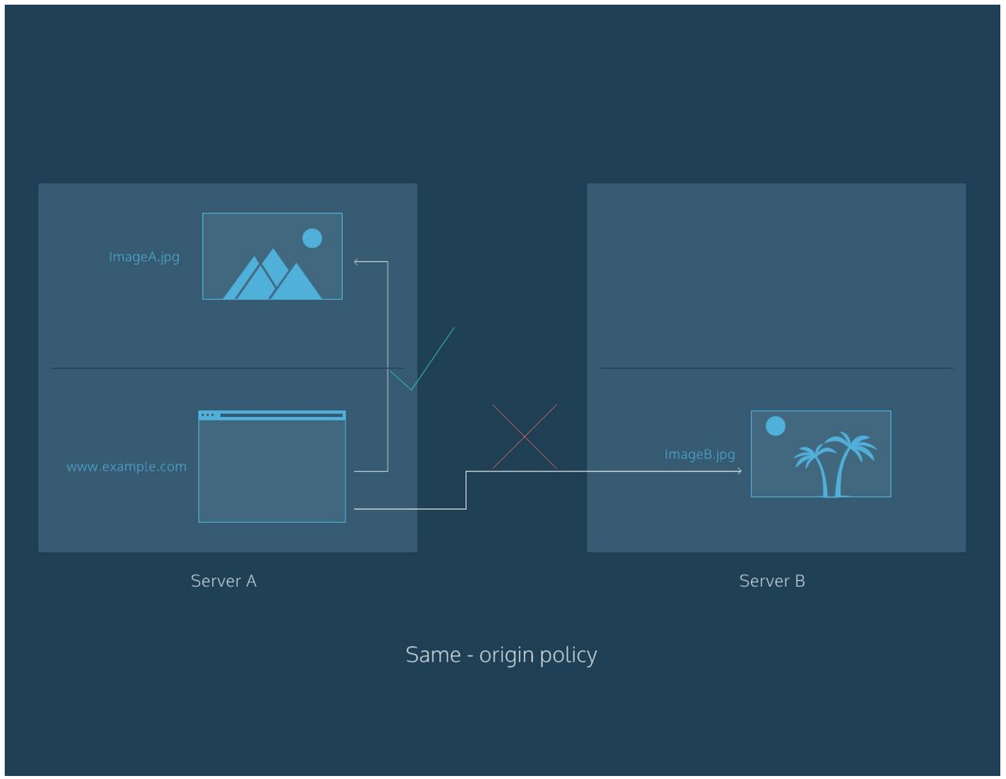


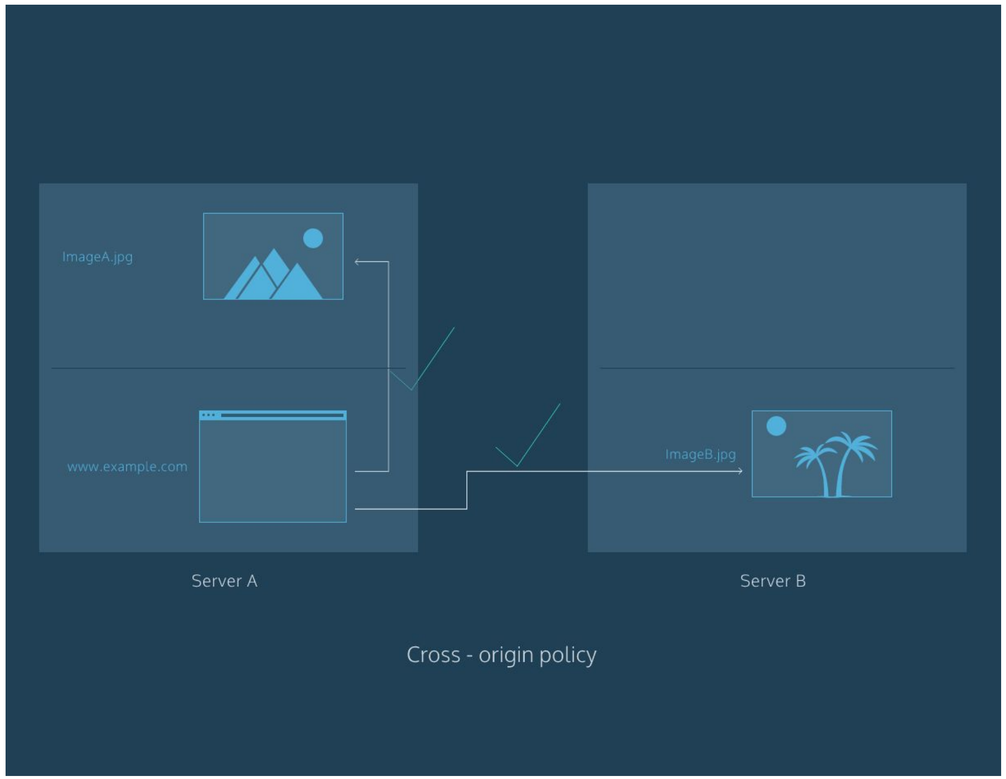

Mode d’emploi

Méthode de requête	URL
POST	http://server.com/api/v1/users
GET	http://server.com/api/v1/users
GET	http://server.com/api/v1/users/1
PUT	http://server.com/api/v1/users/1
DELETE	http://server.com/api/v1/users/1
DELETE	http://server.com/api/v1/users

Mode d’emploi

Cross-origin resource sharing (CORS)










Mode d’emploi

Cross-origin resource sharing (CORS)

HTTP Request

```
GET http://myservice.azurewebsites.net/api/test HTTP/1.1
Accept: */*
Accept-Language: en-US
Origin: http://myclient.azurewebsites.net
User-Agent: Mozilla/5.0
Host: myservice.azurewebsites.net
```






Mode d’emploi

Cross-origin resource sharing (CORS)

HTTP Response

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: text/plain; charset=utf-8
Access-Control-Allow-Origin: http://myclient.azurewebsites.net
Date: Wed, 05 Jun 2013 06:27:30 GMT
Content-Length: 17

GET: Test message
```

CRUD

Les quatres fonctions de base du
stockage persistant.

Mode d’emploi

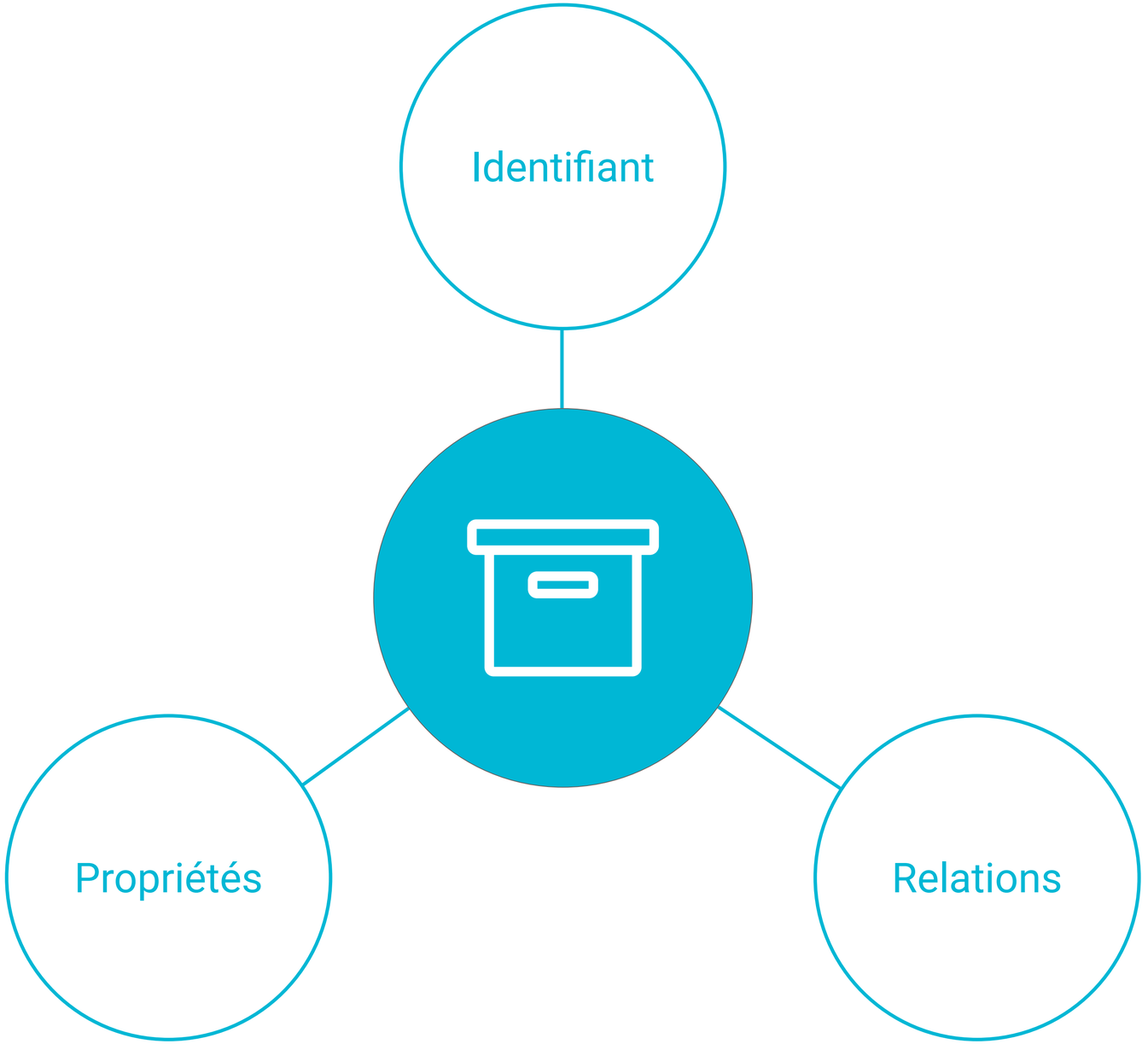
CREATE	READ	UPDATE	DELETE
POST	GET	PUT	DELETE

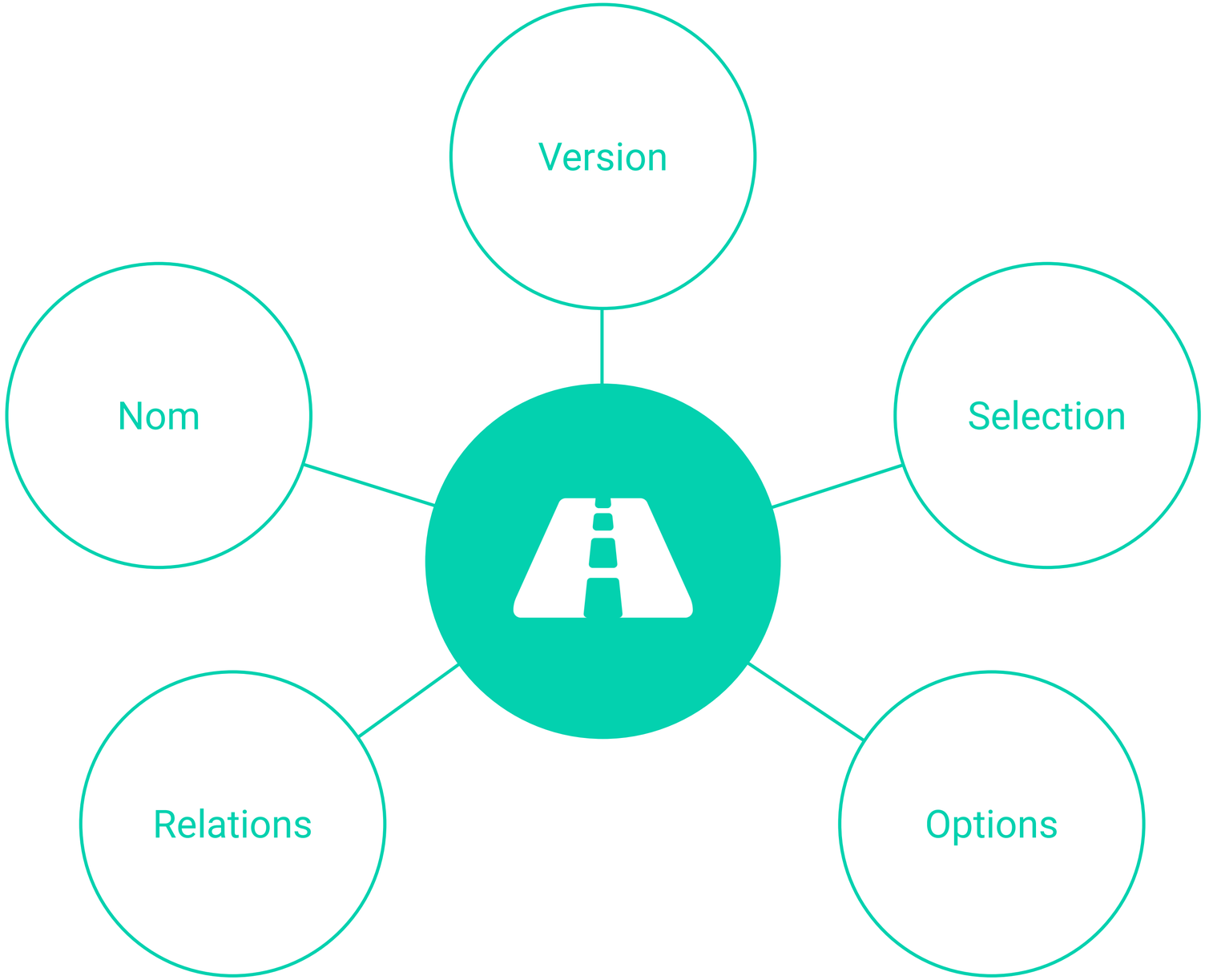
Conception

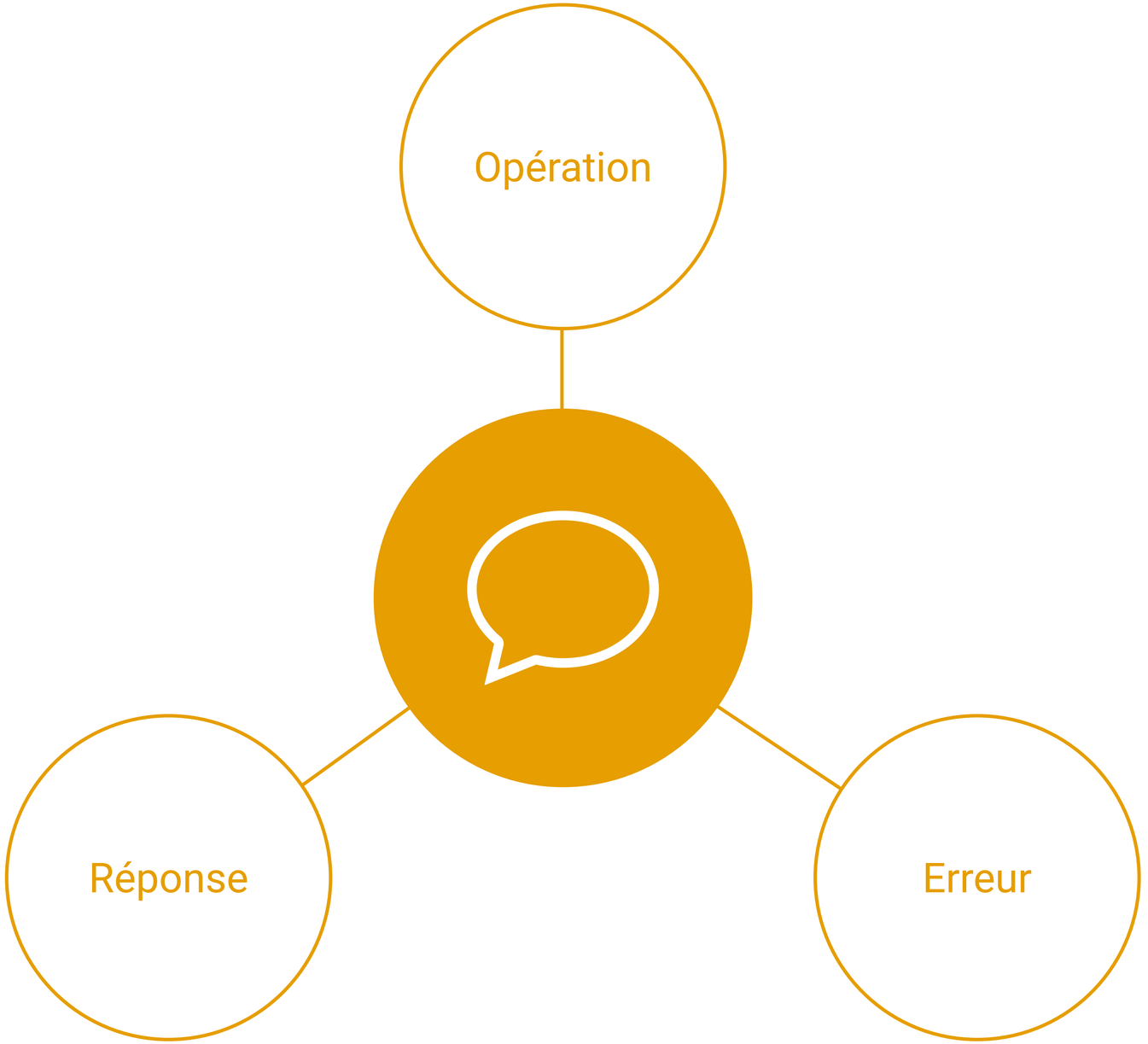
Objet

Route

Action


Identifiant
Propriétés
Relations


Version
Nom
Selection
Relations
Options


Opération
Réponse
Erreur

Conception

Version

Toujours versionner le service pour identifier les fonctionnalités.

Penser aux options

N’oubliez pas les options pour le filtrer, trier, sélectionner et paginer.

Noms uniquement

Que des noms, pas de verbes. Pour éviter une documentation trouble.

Créer les relations

Pensez toujours aux relations entre vos objets.

Noms au pluriel

Si vous ne savez pas quelle forme adopter, utilisez le pluriel.

Conception

Version

http://server.com/api/**v1**/...

http://server.com/api/**v2.1**/...

http://server.com/api/**1.1.0**/...

Conception

Noms uniquement

http://server.com/api/v1/**users**

http://server.com/api/v1/**doctors**/10

~~http://server.com/api/v1/**get-all-users**~~

~~http://server.com/api/v1/**get-user**/10~~

Conception

Noms au pluriel

http://server.com/api/v1/**cars**

http://server.com/api/v1/**drugs**/10

~~http://server.com/api/v1/**joint**/420~~

~~http://server.com/api/v1/**car**~~

Conception

Penser aux options

http://server.com/api/v1/cars?**color=red**&**seats=2**

http://server.com/api/v1/users?**sort=+birthday,-name**

http://server.com/api/v1/doctors?**fields=name,birthday,rpps**

http://server.com/api/v1/cars?**offset=10&limit=5**

Conception

Créer les relations

http://server.com/api/v1/**users**/**10**/**comments**

http://server.com/api/v1/**doctors**/**542**/**patients**?disease=flu

Securité

Hachage de
mot de passe

Pare-feu
applicatif

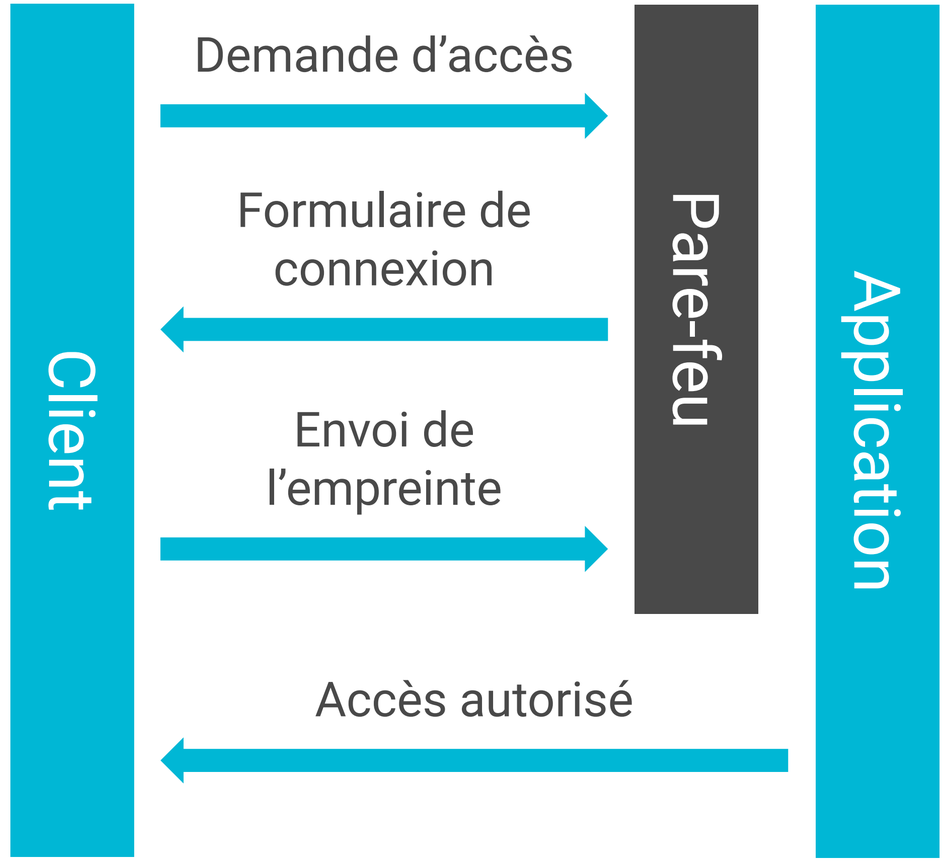
Jeton d’accès

Securité

Algorithme de hachage
SHA-512, Bcrypt, ...

Stockage de l'empreinte
Seulement l'empreinte est stockée.

Comparaison des empreintes
Lors de l'authentification, les empreintes sont comparées.

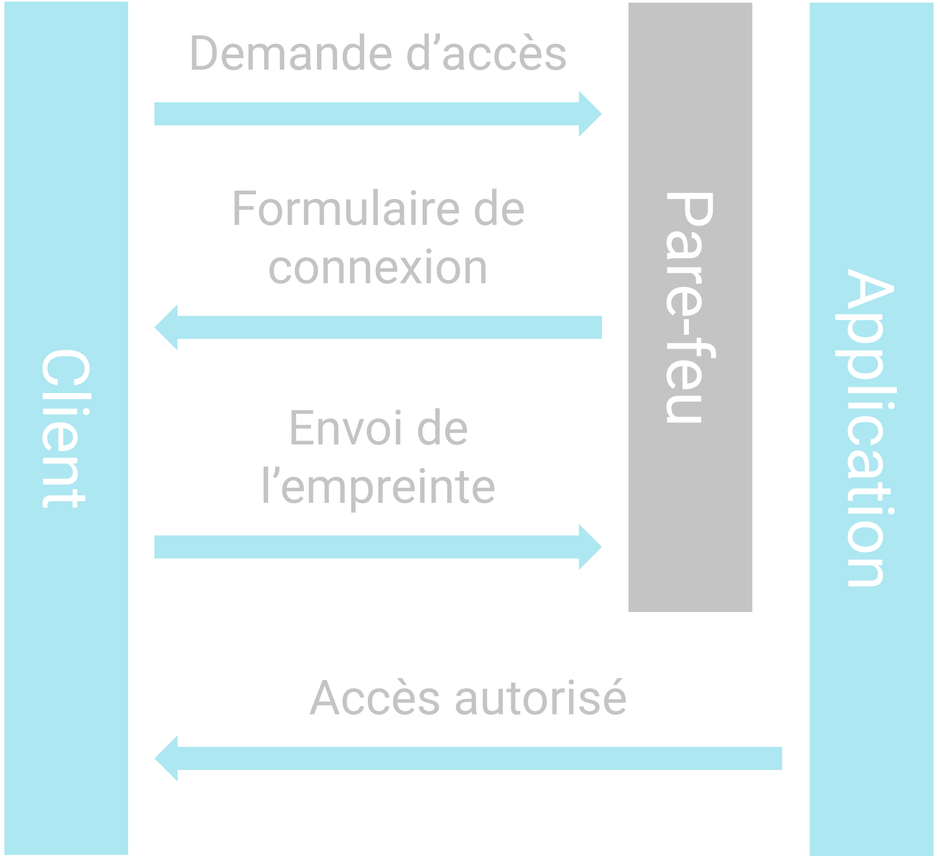


Securité

Une partie de l’application
Le pare-feu fait partie intégrante de l’application.

Connexions entrantes
Le pare-feu analyse toutes les connexions entrantes.

Vérifier les connexions
Le pare-feu vérifie si les jetons d’accès sont correctes.



Securité

Clef API
Une clef API est une chaine de charactère permettant d'identifier un système afin de lui accorder un accès à l'application.

Jeton OAuth2
Permet à un utilisateur de donner l’accès limité aux données de l’API à un service tierce.

Signature HMAC
HMAC est un type de code d'authentification de message (MAC) calculé en utilisant une fonction de hachage en combinaison avec une clé secrète.

Travaux pratiques

1. Installation des outils

2. Création d’un simple service Web

3. Mise en place d’un CRUD

4. Relations et filtres

5. Pare-feu

6. Jeton web JSON