Communiquer avec des Web services

Comment la communication client-serveur marchent-elle?

KINSHASA DIGITAL ACADEMY

https://bit.ly/3fgLLp1

KINSHASA
DIGITAL
ACADEMY

Service oriented architecture (SOA)

KINSHASA DIGITAL ACADEMY

SOA c’est quoi?

L'architecture orientée services (ou SOA, Service-Oriented Architecture) est un modèle de conception qui rend des composants logiciels réutilisables, grâce à des *interfaces* de ***services*** qui utilisent *un langage commun* pour *communiquer via un réseau*. Avant l'arrivée de l'architecture SOA à la fin des années 1990, il était difficile de connecter une application à des services hébergés par un autre système

KINSHASA DIGITAL ACADEMY

SOA : C’est quoi un service

Un service est une *unité autonome de fonctionnalité logicielle*, ou d'un *ensemble de fonctionnalités*, conçue pour réaliser *une tâche précise* comme *récupérer des informations* ou *exécuter une opération*. Vous pouvez y accéder à *distance*, et interagir avec lui ou le mettre à jour de manière *indépendante*.

KINSHASA
DIGITAL
ACADEMY

SOA : Les standards

Parmi les standards basé sur les SOA, il y a :

- ***Le protocol SOAP*** dont le format de communication des données est le XML et qui support plusieurs protocoles(HTTP, SMTP, etc.)
- **L'architecture REST**, qui est un ensemble des principes architecturaux adapté aux besoin des services web et application mobile.Il est basé sur HTTP

KINSHASA DIGITAL ACADEMY

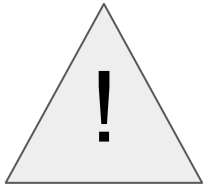
Architecture REST

Acronyme de **R**epresentational **S**tate **T**ransfert , c'est une architecture basée sur le protocol HTTP qui permet de développer des web services.A la réception d'une requête, les API créées sur base de ce standard peuvent envoyer des messages en XML, text brute, HTML en *JSON*(Le format d'échange le plus utilisé). À la différence de SOAP, REST n'est ***pas un Protocol***.

KINSHASA
DIGITAL
ACADEMY

REST : Standard

Une API qui respecte les standard définis par REST est appelée RESTful,car pour être appelé RESTful l'API se doit remplir certaines exigences.



Tout web service basé sur HTTP et qui renvoi des messages en JSON n'est pas forcément RESTful

KINSHASA DIGITAL ACADEMY

REST : Standard

Pour qu'une API soit qualifiées de RESTful, elle doit respecter les règles suivantes:

- Séparation du client et du serveur
- Stateless(Sans état)
- Cachable(Cacheable)
- Layered(Système à plusieurs couches)
- Interface Uniforme

Séparation du client et serveur

Il y a une séparation entre l'application client et l'application serveur.L'application client est entièrement responsable de la couche de présentation et le serveur(application serveur) est responsable de recevoir des requêtes et retourner des données.

Stateless

Stateless ou sans état fait au comportement selon lequel l'application serveur ne garde pas en mémoire des informations des requêtes à l'instar des sessions.Dans ce modèle d'architecture, c'est le client qui est responsable de garder les informations sur les sessions des utilisateur connecté par exemple.

KINSHASA
DIGITAL
ACADEMY

Cacheable

Il s’agit pour l’application cliente de limiter le nombre des requêtes faites à l’API REST. L’application cliente a la responsabilité de garder en cache les données qu’elle reçoit du serveur car dans certains cas selon les données demandées la requête peut être plus longue, et c’est une bonne pratique d’éviter au client d’effectuer des requêtes inutiles.

Il est à noter que le mécanisme de caching dépend de la nature de l’application.

Layered : Système en couche

REST vous permet d'utiliser une architecture de système en couches dans laquelle vous déployez les API sur le serveur A, puis stockez les données sur le serveur B et authentifiez les requêtes sur le serveur C, par exemple. Un client ne peut généralement pas dire s'il est connecté directement au serveur final ou à un intermédiaire en cours de route.

KINSHASA DIGITAL ACADEMY

Interface Uniforme

Lors de la création d’une API REST, les développeurs acceptent d’utiliser les mêmes normes. Ainsi, chaque API a une interface uniforme. L’interface constitue un contrat entre le client et le service, que partagent toutes les API REST. C’est utile, car lorsque les développeurs utilisent des API, cela leur permet d'être sûrs qu’ils se comprennent entre eux.

KINSHASA DIGITAL ACADEMY

Interface Uniforme : Ressource

Cette contrainte est complètement orientée vers les **ressources**.Une **ressource** est tout objet nominal,c'est-à-dire un objet auquel on peut attribuer un nom comme ou une entité que l'on peut manipuler, c'est-à-dire : créer, supprimer, mettre à jour et lire au travers d'une API.

Par exemple sur Facebook, il y a des **utilisateurs** qui représentent la ressource utilisateurs.

KINSHASA DIGITAL ACADEMY

Ressource

Ceci est un carton des **jouets.**En d’autre terme ce cartons contient un collection des jouets, c’est pourquoi **jouets** il est au **pluriel**.

Dans cette **collection** des **jouets** on peut identifier chaque **jouet** de manière individuel

KINSHASA DIGITAL ACADEMY

Ressource

Si on veut prendre tous les jouets,dans le carton, on va demander de sortir **tous** les **jouets**

En d'autre terme, notre demande sera : dans la collection des jouets, **récupérer(GET)** tous les **jouets**

KINSHASA DIGITAL ACADEMY

Ressource

KINSHASA DIGITAL ACADEMY

Si on veut prendre un seul jouet, on dira : Dans la collection des **jouets**, prends**(GET)** moi le jouet identifié par **#2** ou **#gir** pour giraffe.

Avec **#gir**, on peut avoir plusieurs **giraffe**, et donc une bonne façon de nommer peut être l'attribution des identifiants uniques, comme par exemple des **nombres**.De cette manière on peut le remplacer**(PUT/PATCH)** s'il est abimé ou le retirer(**DELETE**) de la collection

Ressource : Création

Si on veut , on peut aussi **ajouter(POST)** un nouveau jouet dans notre **collection** des **jouets.**

KINSHASA DIGITAL ACADEMY

Ressource

KINSHASA DIGITAL ACADEMY

Dans cette on collection des jouets, nous pouvons donc:

- Tout **recuperer** : **GET** /jouets
- Récupérer un seul : **GET** /jouets/**2**
- Supprimer un seul : **DELETE** /jouets/**2**
- Modifier un seul : **PUT/PATCH** /jouets/**2**
- Ajouter un jouet : **POST** /jouets

POST, **GET**, **PUT/PATCH**,**DELETE** sont ce que l'on appel des **verbes HTTP** qui nous permettent de manipuler les ressources via des **requêtes HTTP**.

Ressource : contraintes

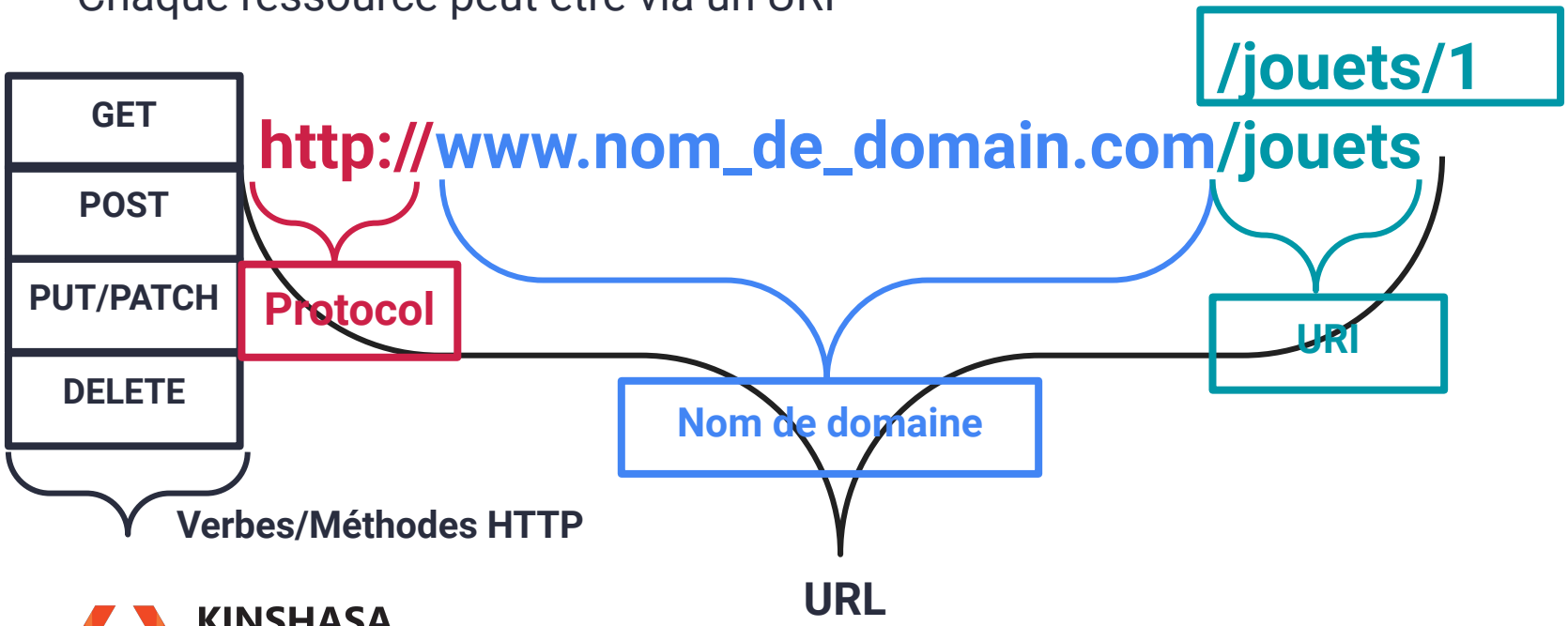
Une ressource doit respecter les contraintes suivantes:

- Posséder un identifiant unique
- Avoir une représentation
- Être auto-décrite.

KINSHASA
DIGITAL
ACADEMY

Ressource

Chaque ressource peut être via un URI



KINSHASA DIGITAL ACADEMY

Ressource : Representation

Une ressource doit avoir une représentation qui doit être dans un format standard compréhensible par les 2 entités qui communiquent.Le formation qui est actuellement le plus utilisé est JSON.

https://eager-murdock-da3a34.netlify.app/

KINSHASA DIGITAL ACADEMY

Ressource : Representation JSON

```json
{
"id": 2,
"nom": "Matungulu",
"prenom": "Johnny",
"age": 23,
"parents": {
    "pere": "Matungulu Madesu Fimbu",
    "mere": "Limbondo Bitabe"
},
"freres": ["Jean-Dindon Matolo", "Guy-Mungwa Makayabu"]
}
```

KINSHASA DIGITAL ACADEMY

Ressource : Representation XML

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<root>
<id>2</id>
<nom>Matungulu</nom>
<prenom>Johnny</prenom>
<age>23</age>
<parents>
    <pere>Matungulu Madesu Fimbu</pere>
    <mere>Limbondo Bitabe</mere>
</parents>
<freres>Jean-DIndo Matolo</freres>
<freres>Guy-Mungwa Makayabu</freres>
</root>
```

KINSHASA DIGITAL ACADEMY

Dialoguer avec les webservices

Pour communiquer avec les web services, nous allons utiliser un client HTTP, qui est un composant logiciel exécuté par l'application cliente afin d'effectuer des requêtes HTTP vers un serveur WEB.

Dans le monde du développement web il y a plusieurs API à l'instar de **fetch** ou **XMLHttpRequest(XHR)** qui sont intégré au navigateur et d'autres à l'exemple de **Axios** qui sont des packages tiers.Ces client font ce que l'on appel des appels **AJAX.**

KINSHASA DIGITAL ACADEMY

AJAX

C’est une approche qui fait en sorte que les applications Web soient capables de réaliser des mises à jour rapides et **incrémentielles** de **l'interface utilisateur** sans **devoir recharger la page entière du navigateur**. Les applications fonctionnent plus rapidement et sont plus **réactives** aux actions de l'utilisateur.

KINSHASA
DIGITAL
ACADEMY

AJAX : Utilisation de fetch()

```
fetch("http://jsonplaceholder.typicode.com/users")
  .then(function (res) {
    return res.json();
  })
  .then(function (data) {
    //Vos données sont disponibles ici dans data
    //Vous pouvez en faire ce que vous voulez
  });
```

KINSHASA DIGITAL ACADEMY

Écrivons du code 😋

KINSHASA
DIGITAL
ACADEMY