esprit
Se former autrement
HONORIS UNITED UNIVERSITIES

# Services Web REST

**Module SOA A.U 2024-2025**

United Nations Educational, Scientific and Cultural Organization
UNESCO Chair "Project-based learning" ESPRIT School of engineering, Tunisia
EUR-ACE® Délivrée par la Commission des Titres d'Ingénieur
CONFÉRENCE DES GRANDES ÉCOLES
cdio CONCEIVE DESIGN IMPLEMENT OPERATE

# Objectifs

- Comprendre le style d'architecture REST.
- Concevoir et consommer des services Web RESTful

# Plan

- Présentation de SW REST
- Motivation pour REST
- Principes de REST
- Architecture RESTful
- Contrat WADL

# Présentation de REST 1/2

- REST est l'acronyme de **REpresentational State Transfert**

- Principe défini dans la thèse de Roy FIELDING en 2000

  -L'un des principaux auteurs de la spécification HTTP

  -Le développeur du serveur Web Apache

- REST est un **style d'architecture** inspiré de l'architecture du **Web** pour construire des services web

- Les applications respectant les architectures orientées ressources sont nommées **RESTful**

Un style d'architecture est un ensemble de contraintes qui permettent, lorsqu'elles sont appliquées aux composants d'une architecture, d'optimiser certains critères propres au cahier des charges du système à concevoir.

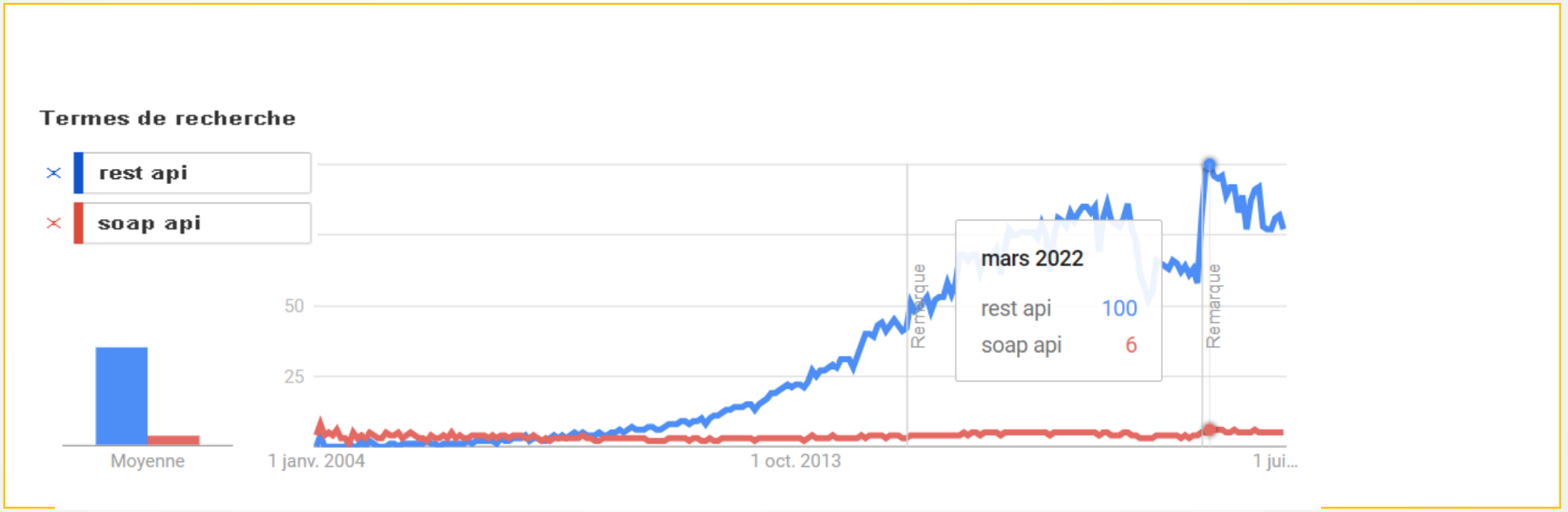
# Présentation de REST 2/2

- REST est:
  - un style d'architecture non standardisé
  - une approche pour construire une application
- REST n'est pas:
  - un format
  - un protocole
  - un standard
- Bien que REST ne soit pas un standard, il utilise des standards:
  - HTTP
  - URL
  - XML/HTML

# Motivation pour REST 1/2

- REST est une alternative à SOAP

- En 2006, Google a abandonné son API SOAP au profit d'une API simplifiée REST




**Source:** https://trends.google.com/trends/explore?date=all&q=rest%20api,soap%20api&hl=fr

# Motivation pour REST 2/2

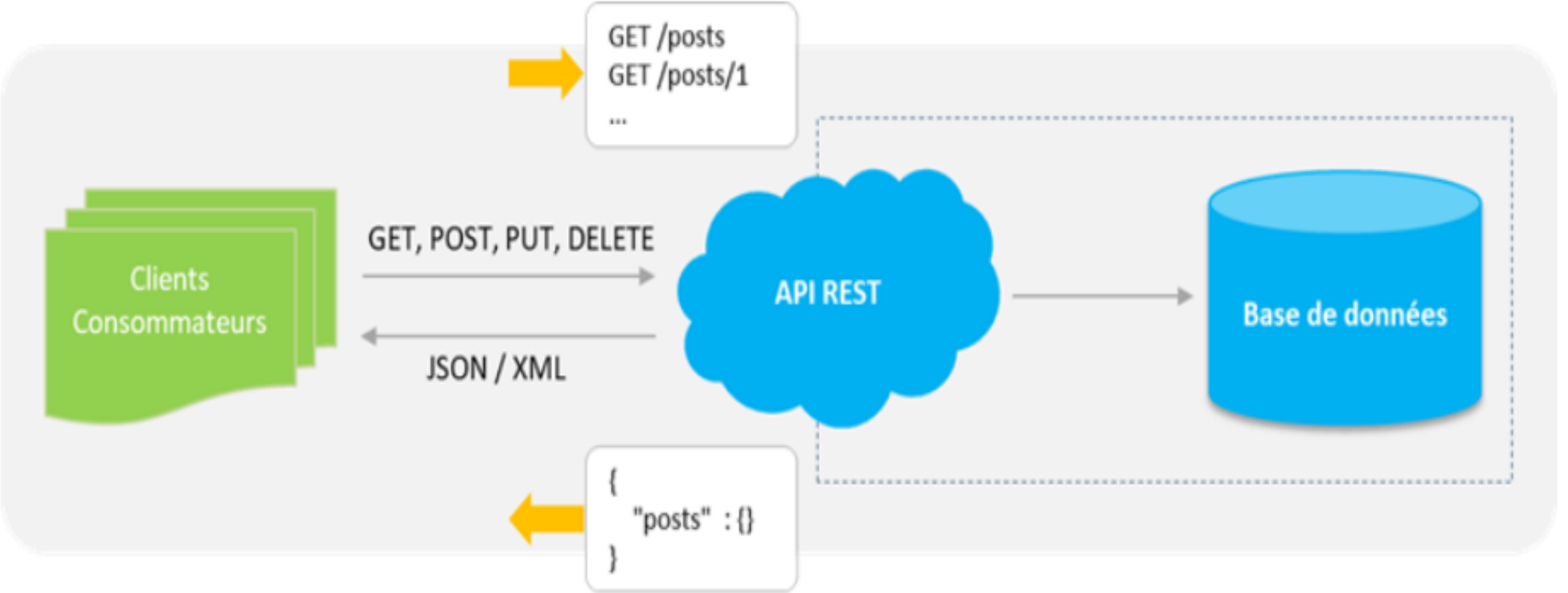
- REST est léger et simple :
  Les messages sont courts, faciles à décoder par le navigateur et par le serveur d'application.

- REST est auto-descriptif :
  Vous pouvez naviguer à travers ses ressources comme vous le feriez avec une page Web. Il y a une URL intuitive unique pour chaque ressource. On peut facilement en déduire la structure des ressources sans avoir besoin de beaucoup de documentation.

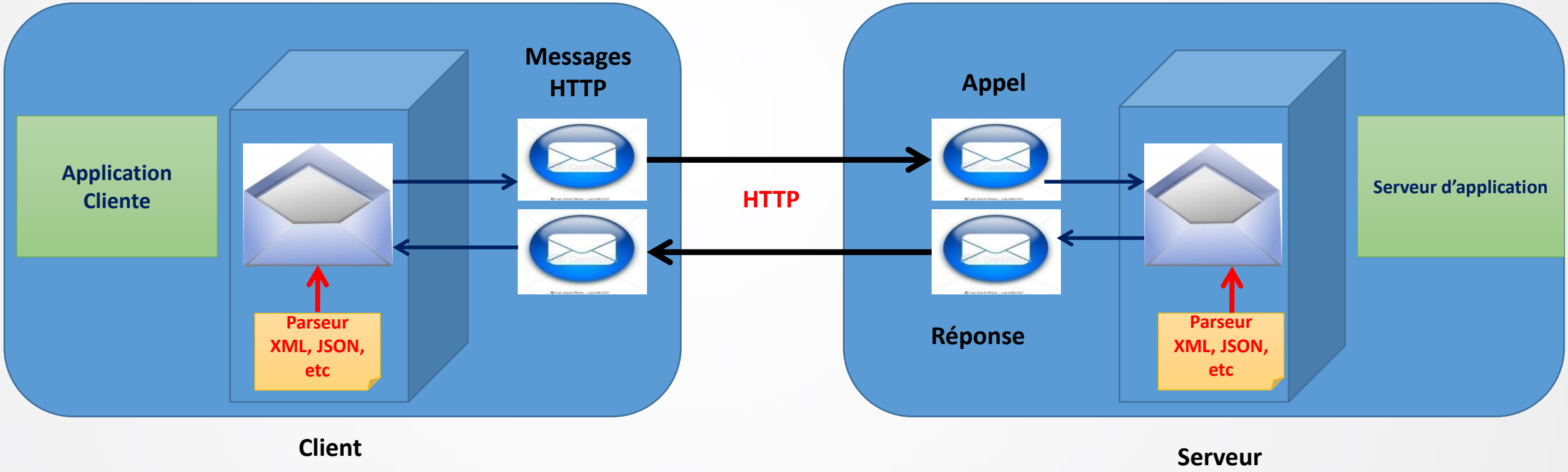
- REST est stateless :
  Consommation de mémoire inférieure

- REST peut être géré en cache
  Mise en cache possible donc meilleure montée en charge

# Architecture RESTful



# Fonctionnement d'un Service Web REST



# Principes de REST 1/7

**URI**

http://weather.com/tunis

Identifie

**Ressource**

La météo de Tunis


Représente

**Représentation**

```
Metadata:
Content-type:
application/xhtml+xml
---------------------
Data:
<!DOCTYPE html PUBLIC "...
    "http://www.w3.org/...
<html xmlns="http://www...
<head>
<title>5 Day Forecaste for
Oaxaca</title>
...
</html>
```

# Principes de REST 2/7

- Une ressource
- Un identifiant de ressource
- Une représentation de la ressource
- Interagir avec les ressources
  - Requêtes HTTP : GET, POST, PUT et DELETE

# Principes de REST 3/7

- **Ressources (Identifiant)**
  - Identifié par une URI

    Exemple : http://localhost:8080/libraryrestwebservice/books

**Méthodes (Verbes)**

- Pour manipuler la ressource
- Méthodes HTTP : GET, POST, PUT and DELETE

- **Représentation**
  - Donne une vue sur l'état de la ressource
  - Informations transférées entre le client et le serveur

    Exemples : XML, Text, JSON, …

# Principes de REST 4/7

## ❑ Méthodes

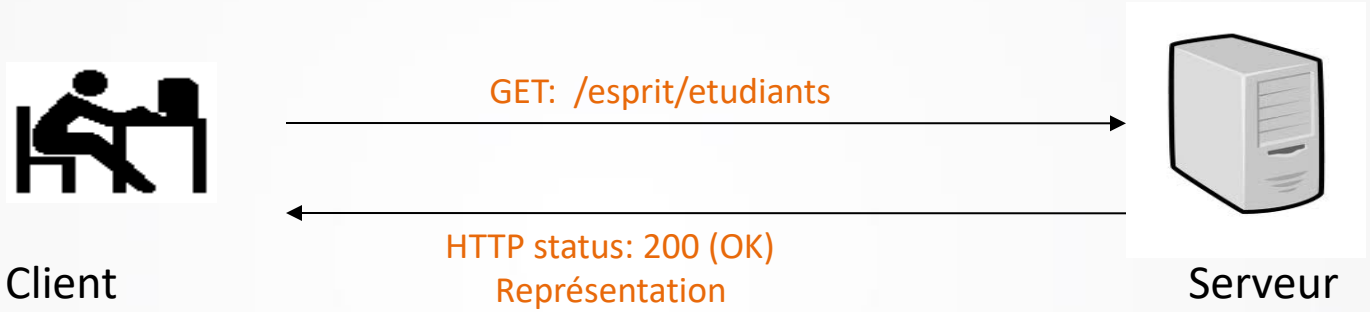
- Une ressource quelconque peut subir quatre **opérations** de base désignées par **CRUD:**
  - **C**reate (Créer)
  - **R**etrieve (Lire)
  - **U**pdate (mettre à jour)
  - **D**elete (Supprimer)

- REST s'appuie sur le protocole **HTTP** pour exprimer les opérations via les méthodes HTTP
  - Create ⟷ **POST**
  - Retrieve ⟷ **GET**
  - Update ⟷ **PUT**
  - Delete ⟷ **DELETE**

# Principes de REST 5/7

## ❑ Méthodes

- Méthode **GET** fournit la représentation de la ressource




- Méthode **POST** crée une ressource

# Principes de REST 6/7

## ❑ Représentation

Désigne les données échangées entre le client et le serveur pour une ressource:

- le client (GET): format de sortie
- le serveur (PUT et POST): format d'entrée

La représentation d'une ressource peut prendre différents formats:

- XML
- JSON
- Text, HTML
- …

Le format d'entrée (PUT et POST) et le format de sortie (GET) d'un service Web d'une ressource peuvent être différents.

# Principes de REST 7/7

## ❑ Représentation

Désigne les données échangées entre le client et le serveur pour une ressource:

- le client (GET): format de sortie
- le serveur (PUT et POST): format d'entrée

La représentation d'une ressource peut prendre différents formats:

- XML
- JSON
- Text, HTML
- …

Le format d'entrée (PUT et POST) et le format de sortie (GET) d'un service Web d'une ressource peuvent être différents.

# WADL 1/2

- **W**eb **A**pplication **D**escription **L**anguage
- Un langage de description XML de services de type REST
- Une spécification W3C initiée par SUN
- l'objectif est de pouvoir générer automatiquement les APIs clientes d'accès aux services REST

*Remarques*

-Peu d'outils exploite la description WADL

-Apparu bien plus tard

# WADL 2/2

Exemple

```
<application>
<doc jersey:generatedBy="Jersey: 1.4 09/11/2010 10:30 PM"/>
<resources base="http://localhost:8088/librarycontentrestwebservice/">
        <resource path="/contentbooks">
                <resource path="uribuilder2">
                        <method name="POST" id="createURIBooks">
                                <request>
                                        <representation mediaType="application/xml"/>
                                </request>
                                <response>
                                        <representation mediaType="*/*"/>
                                </response>
                        </method>
                </resource>
                <resource path="uribuilder1">
                        <method name="POST" id="createBooksFromURI">
                                <request>
                                        <representation mediaType="application/xml"/>
                                </request>
                                <response>
                                        <representation mediaType="*/*"/>
                                </response>
                        </method>
                </resource>
                ...
        </resource>
</resources>
</application>
```

# Développement d'un Service Web REST

- Seule **l'approche bottom-up** est possible:
  - Implémenter l'application.
  - Compiler, déployer et tester.
  - Une interface de description (Swagger, WADL, etc) peut être générée.

# En résumé

- REST est un style d'architecture
- REST est une alternative aux services web étendus (SOAP)
- REST se base sur le protocole HTTP