

UNIVERSITE ASSANE SECK DE ZIGUINCHOR

UFR: SCIENCES ET TECHNOLOGIES

DEPARTEMENT: INFORMATIQUE

MASTER 1

MATIERE: Web Service

Année académie 2023-2024

**EXPOSANTS:**

Mouhamed Nasssour Cherif KANE

Dieydy DEMBELE

**Sous la direction:**

Dr Camir MALAK

**Sujet:**

RPC( Remote Procedure Call) [JSON-RPC]

Table des matières

[I. Introduction du Web Service 1](#_Toc198460844)

[II. Fonctionnement du Web Service JSON-RPC 1](#_Toc198460845)

[III. Spécification ou Langage de Définition de Données (IDL) : 2](#_Toc198460846)

[IV. Avantages et Inconvénients du Web Service JSON-RPC 3](#_Toc198460847)

[1. Avantages : 3](#_Toc198460848)

[2. Inconvénients 3](#_Toc198460849)

[V. Sécurité du Web Service 3](#_Toc198460850)

[VI. Tableau comparatif des Web Services 3](#_Toc198460851)

[VII. Conclusion : 3](#_Toc198460852)

# ****Introduction du Web Service****

**Définition** : Un **Web Service** est une application accessible via le réseau (comme Internet) qui permet l’échange de données entre systèmes, souvent de manière interopérable et indépendante du langage de programmation.

**Types de Web Services** :

**SOAP** (Simple Object Access Protocol)

**REST** (Representational State Transfer)

**JSON-RPC** / **XML-RPC** (Remote Procedure Call via JSON ou XML)

**JSON-RPC** :

Protocole léger basé sur **JSON**.

Utilisé pour faire des appels de procédures distants (**Remote Procedure Calls**) en encodant les requêtes et les réponses en JSON.

# ****Fonctionnement du Web Service JSON-RPC****

* **Format d’échange de données** : JSON (JavaScript Object Notation)
* **Principe** :
* Un client envoie une requête JSON au serveur spécifiant :
* Le nom de la méthode à invoquer (method)
* Les paramètres (params)
* Un identifiant (id)
* Le serveur traite l’appel et renvoie une réponse JSON avec :
* Le résultat (result)
* Le même identifiant (id)
* Ou une erreur (error)

**Exemple de requête :**

{

"jsonrpc": "2.0",

"method": "addPerson",

"params": {

"name": "Alice",

"age": 25

},

"id": 1

}

**Exemple de réponse :**

{

"jsonrpc": "2.0",

"result": "Person added successfully",

"id": 1

}

# Spécification ou Langage de Définition de Données (IDL) :

* JSON-RPC **n’impose pas un IDL formel** comme WSDL (SOAP) ou OpenAPI (REST).
* Cependant, pour documenter et décrire les services, on peut utiliser :
* **JSON Schema** pour décrire les structures de données JSON.
* Des outils comme **OpenRPC** (similaire à OpenAPI mais pour JSON-RPC).
* Autres IDL (en comparaison) :
* **WSDL** : SOAP
* **OpenAPI** (anciennement Swagger) : REST
* **Protocol Buffers (Protobuf)** : gRPC

# Avantages et Inconvénients du Web Service JSON-RPC

## Avantages :

* Simple, léger (grâce au format JSON)
* Lisible et facile à déboguer
* Fonctionne sur HTTP, WebSocket, etc.
* Adapté aux systèmes distribués légers ou microservices

1. Inconvénients :

* Moins standardisé que SOAP ou REST
* Pas d’IDl obligatoire → Documentation manuelle
* Peu de support natif dans certains langages/frameworks
* Moins d’outils que REST/OpenAPI pour la génération automatique de code

# Sécurité du Web Service

* JSON-RPC **ne définit pas lui-même la sécurité**.
* La sécurité dépend du **transport utilisé** (souvent HTTP ou WebSocket).
* Mécanismes de sécurité à implémenter :
* **HTTPS (TLS)** : chiffrement des échanges
* **Authentification** : JWT, OAuth, API Key
* **Contrôle d’accès** (ACL, RBAC)
* **Validation des entrées** (contre injection, attaques XSS, etc.)

# Tableau comparatif des Web Services

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Critère** | **JSON-RPC** | **REST** | **SOAP** | **gRPC** | **GraphQL** |
| **Format d’échange** | JSON | JSON / XML | XML | Protobuf (binaire) | JSON |
| **Transport** | HTTP, WebSocket | HTTP | HTTP, SMTP | HTTP/2 | HTTP |
| **IDL** | OpenRPC (optionnel) | OpenAPI | WSDL | .proto (Protobuf) | SDL (Schema Definition) |
| **Lisibilité** | Haute | Haute | Faible | Faible (binaire) | Haute |
| **Poids des messages** | Léger | Moyen | Lourd | Très léger | Moyen |
| **Performance** | Moyenne | Bonne | Faible | Excellente | Bonne |
| **Outils & support** | Limité | Large | Mature | Bon (Google, cloud, etc.) | Large (Facebook, outils) |
| **Sécurité intégrée** | À implémenter | À implémenter | WS-Security disponible | TLS, Auth via metadata | À implémenter |
| **Facilité de documentation** | Moyenne | Excellente (Swagger) | Moyenne | Bonne | Bonne (via introspection) |
| **Requêtes dynamiques** | Non | Partiellement (QueryParams) | Non | Non | Oui (requêtes flexibles) |
| **Sur-fetching / Under-fetching** | Oui | Oui | Oui | Non | Non |

# Conclusion :

Les Web Services jouent un rôle fondamental dans le développement des systèmes distribués et des architectures orientées services. Dans ce contexte, **JSON-RPC** se distingue comme un protocole léger, simple et efficace pour effectuer des appels de procédures distants en utilisant le format JSON.

Grâce à sa **simplicité**, JSON-RPC est adapté aux cas où la performance, la rapidité d’implémentation et l'interopérabilité sont prioritaires. Toutefois, en l’absence de normes strictes pour la documentation et la sécurité, il nécessite une vigilance particulière de la part des développeurs.

Comparé à d'autres solutions comme **SOAP**, **REST**, **gRPC** ou **GraphQL**, JSON-RPC trouve sa place dans des scénarios où la surcharge des standards n’est pas justifiée, notamment dans les microservices, les applications embarquées ou les communications temps réel via WebSocket.

En somme, **le choix d’un Web Service** dépend fortement des **besoins du projet** : complexité, performance, sécurité, et niveau d’interopérabilité. JSON-RPC est une option intéressante à considérer lorsqu’on recherche un compromis entre efficacité et simplicité.