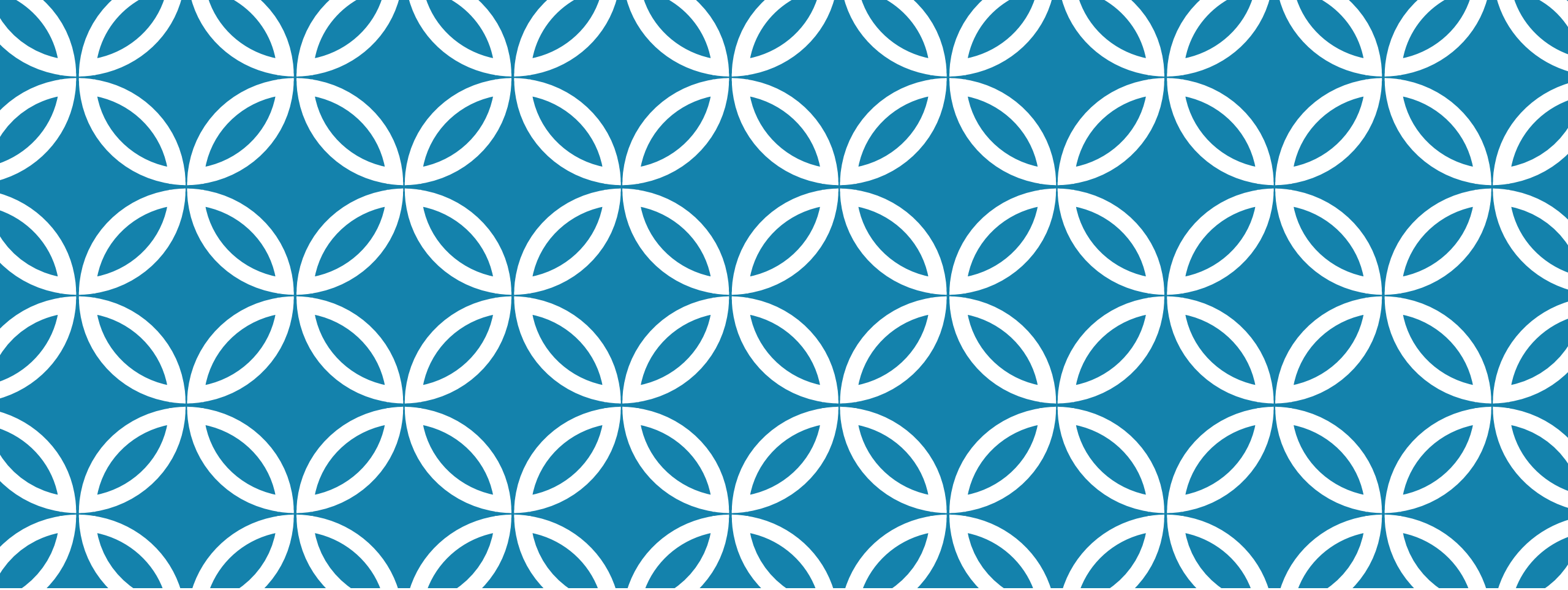
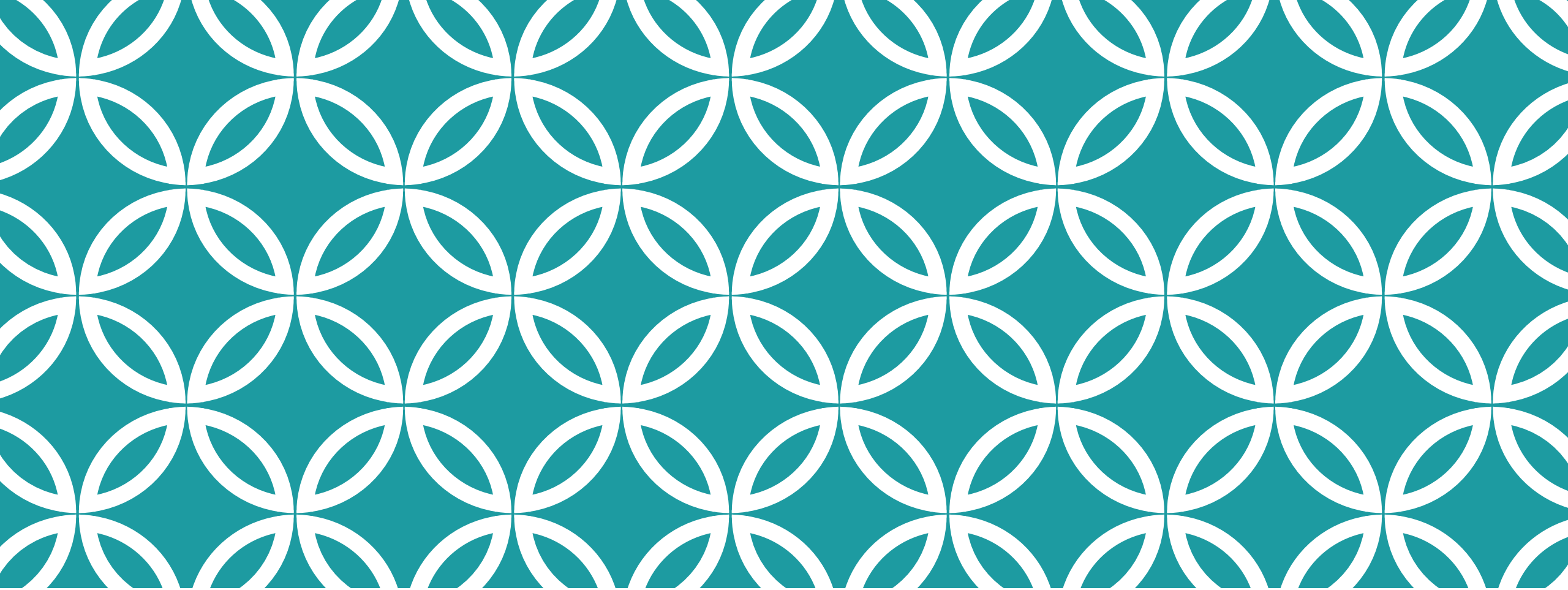


WEB SERVICES

Jérémy PERROUAULT



INTRODUCTION

DÉFINITION

Application logicielle ou composant qui peut échanger via Internet avec n'importe quelle plateforme et langage informatique des données interopérables en utilisant un modèle d'échange de messages rigoureusement défini, et qui peuvent être utilisés par l'application.

Les services web sont

- Des composants métiers
- Auto suffisants
- Auto descriptifs
- Exécutés sur Internet
- Respectant des contrats de qualité

POURQUOI ?

Problématiques d'interopérabilité (fusion ou acquisition d'entreprises)

L'arrivée en 1996 de XML provoque un engouement

Le protocole RPC (**R**emote **P**rocedure **C**all) y répond

- Java RMI
- XML-RPC

POURQUOI ?

Problématiques d'interopérabilité (fusion ou acquisition d'entreprises)

L'arrivée en 1996 de XML provoque un engouement

Le protocole RPC (**R**emote **P**rocedure **C**all) y répond

- Java RMI
- XML-RPC

POURQUOI ?

RPC est une technologie qui a fait des petits

- XML-RPC puis SOAP
- REST

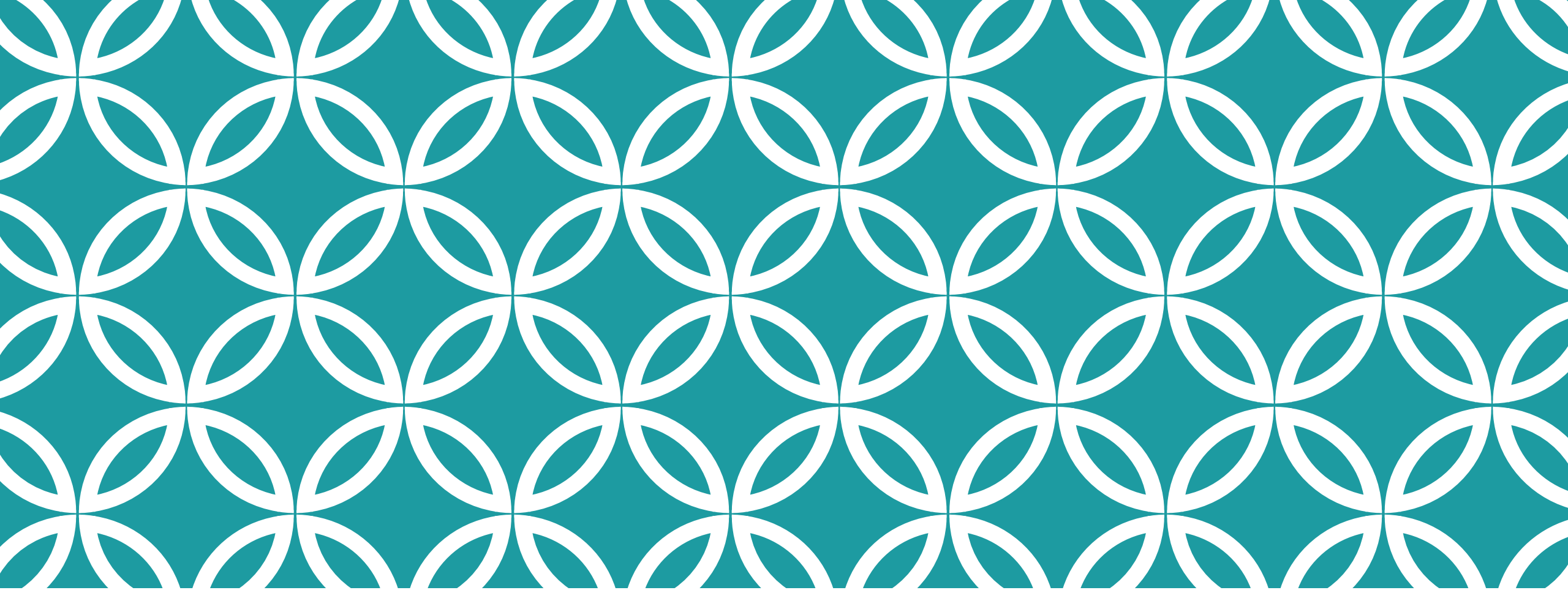
Les principes sont similaires, mais les approchent peuvent être différentes

- Un client demande à un serveur
- Dans le cas de RPC ou SOAP
  - Le client demande une procédure ou une méthode
  - Les entités sont masquées
- Dans le cas de REST
  - Le client demande une ressource / entité
  - Les procédures sont masquées

COMMENT ?

On peut apposer ces techniques d'échange par deux architectures

- ROA (Resource Oriented Architecture)
  - Exposition des ressources (CRUD par exemple)
  - N'induit pas nécessairement **REST**
- SOA (Service Oriented Architecture)
  - Exposition des services (liens entre modules par exemple - microservice)
  - N'induit pas nécessairement **SOAP**



ROA / SOA

ROA

Resource Oriented Architecture

Orienté « ressources » (Internet)

Simple

URL / URI (identifiants des ressources)

ROA – SANS ÉTAT

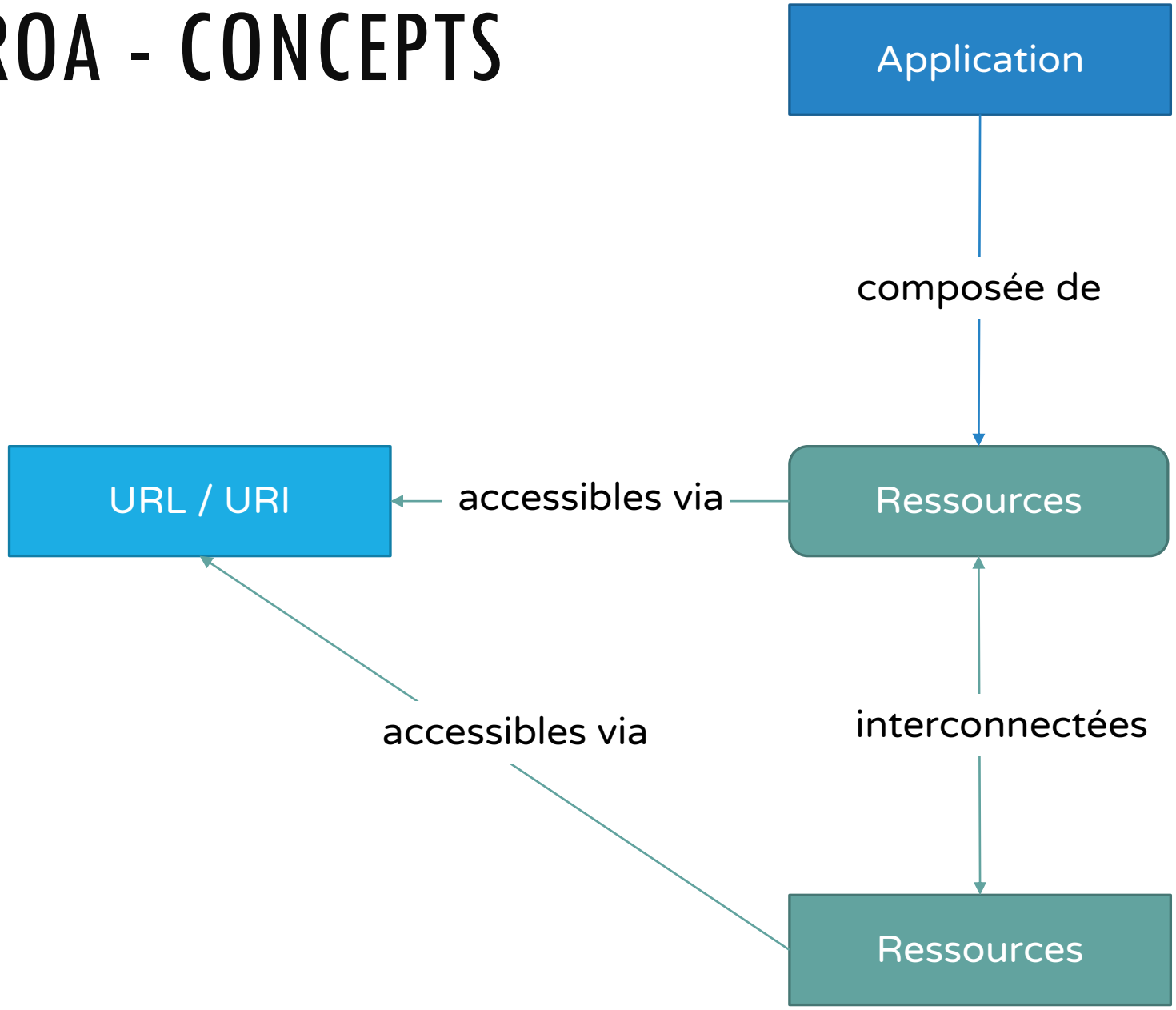
Chaque requête doit s'exécuter sans avoir connaissance des requêtes passées et futures

Les requêtes sont déconnectées les unes des autres

Simple à comprendre : une ressource ne change pas d'état d'une requête à l'autre, elle reste identique (idempotence)

Un état peut être donné à un client

ROA - CONCEPTS



SOA

Services Oriented Architecture

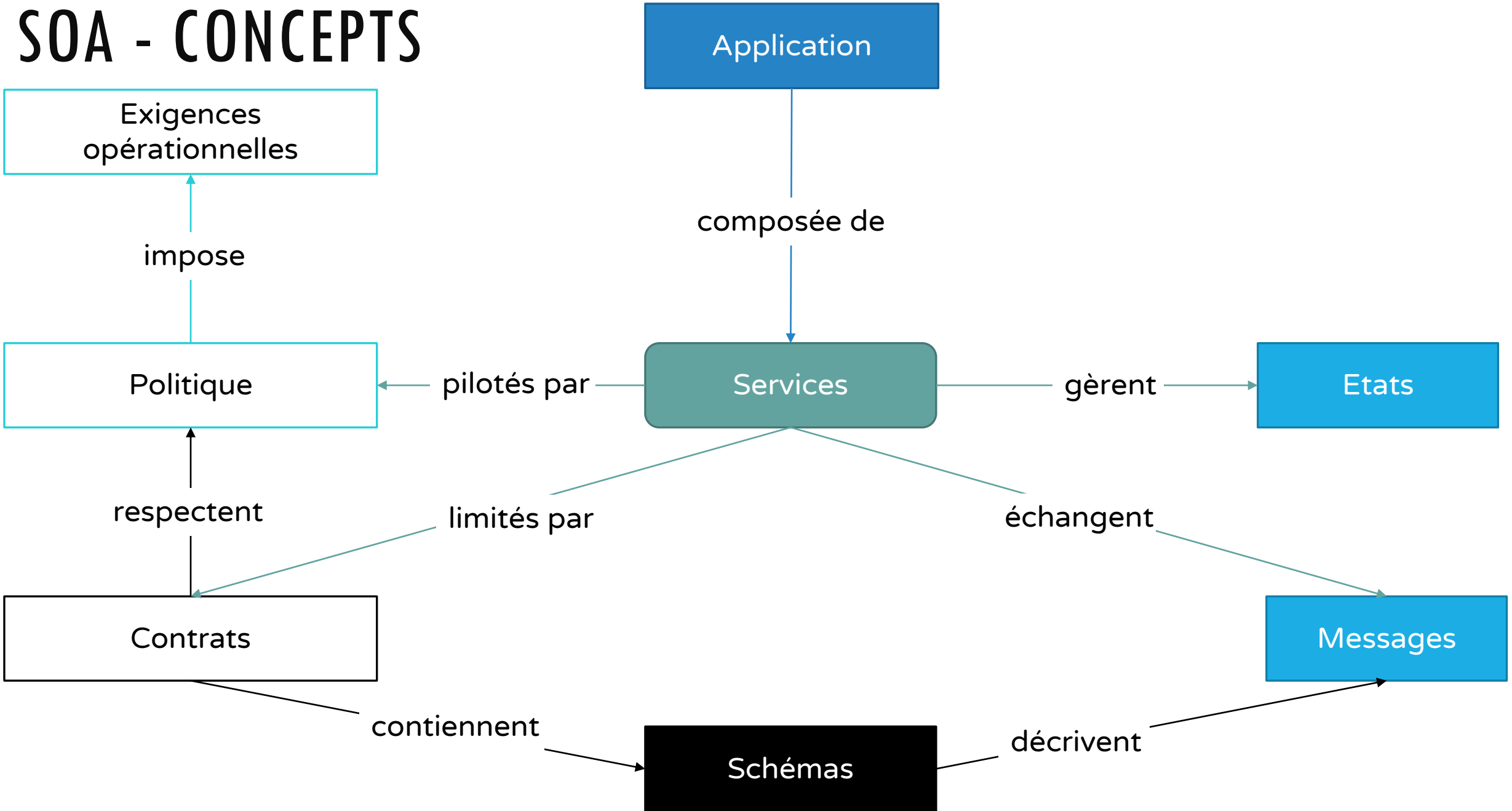
Orienté « Services »

Complexe

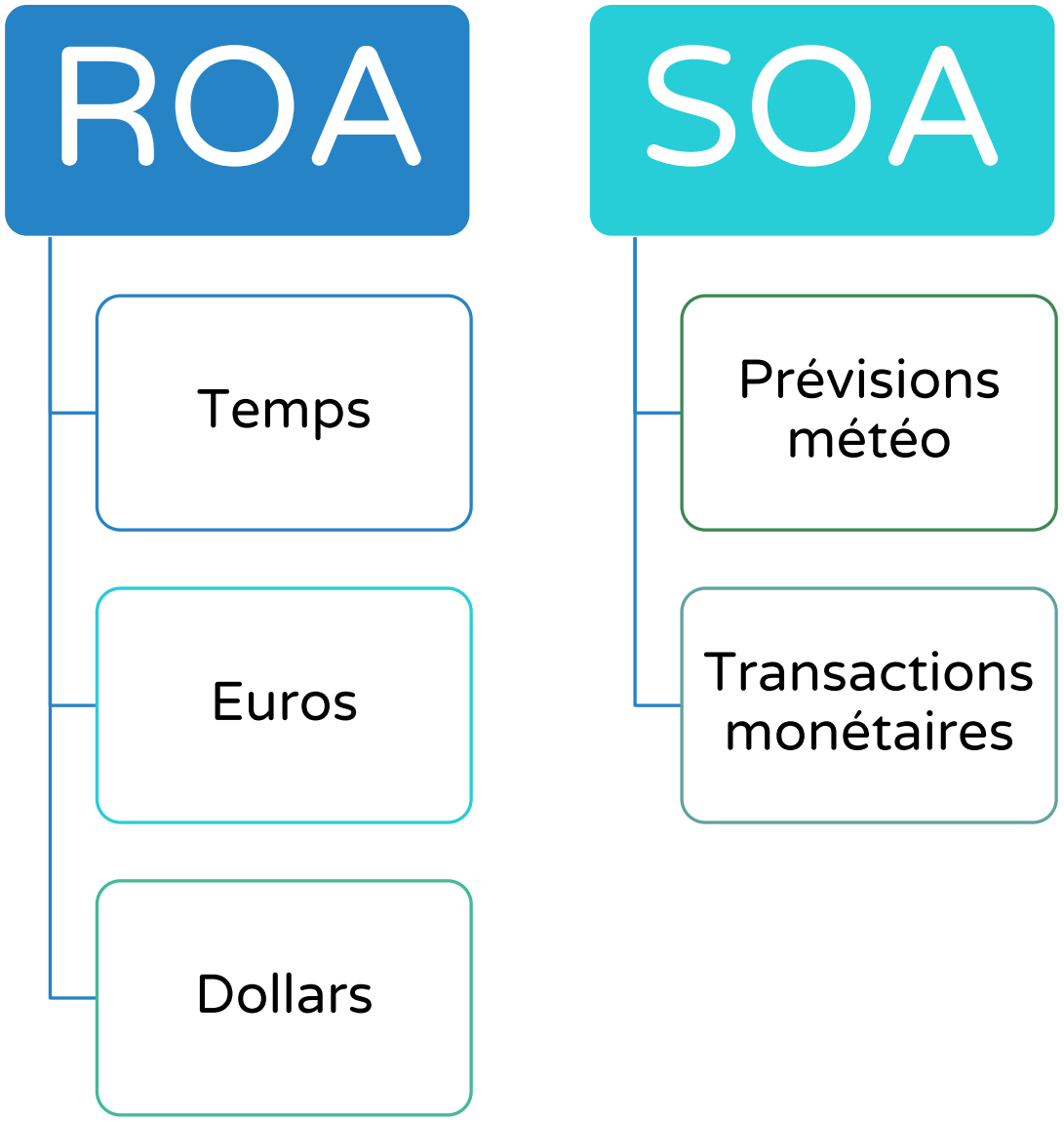
Conservation d’états

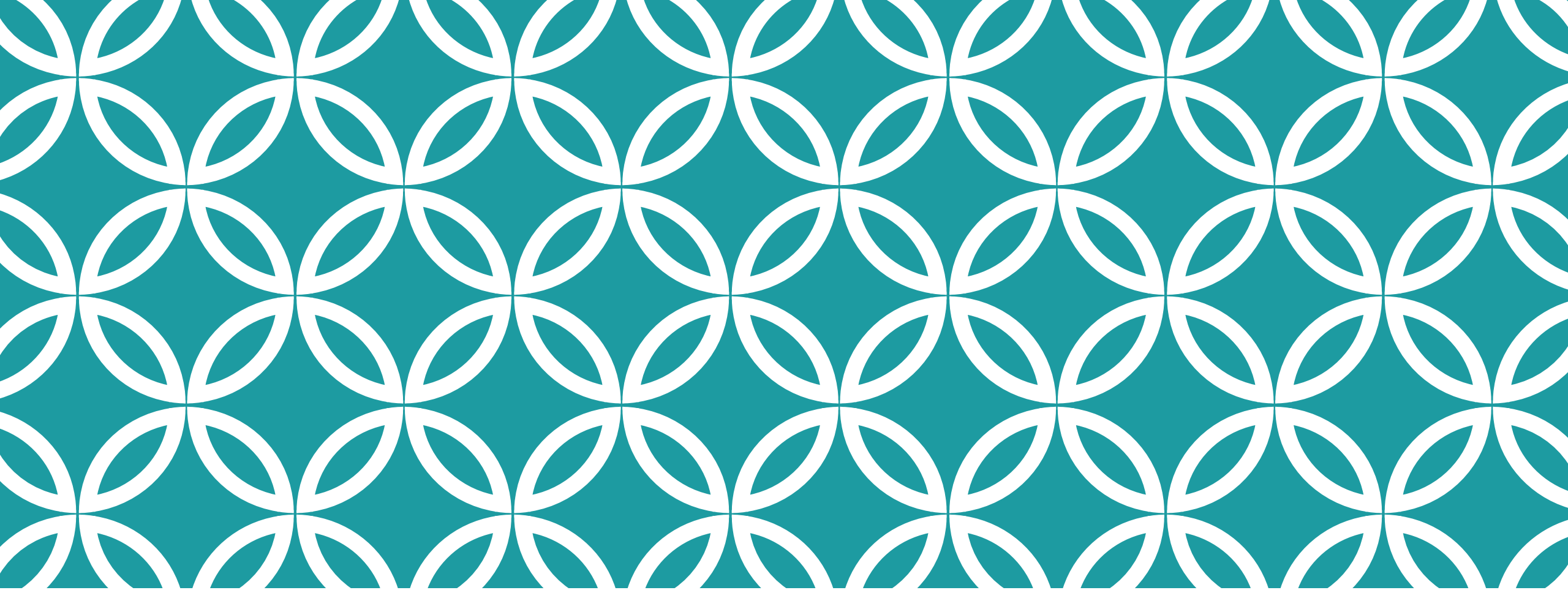
Gestion de contextes

SOA - CONCEPTS



ROA VS SOA


ROA
Temps
Euros
Dollars
SOA
Prévisions météo
Transactions monétaires



SERVICES WEB

REST ou SOAP

SERVICES WEB

Qu’est-ce qu’un service ?

SOAP

REST

REST / HATEOAS

LE SERVICE

Fonctionnalité bien définie

Indépendant

Large granularité

Interface

Localisable

Instance unique (Singleton)

Couplage faible

Synchrone ou asynchrone

CONCEPT

Encapsulation des services

Contrat de service (documents de description)

Abstraction des services dissimulant la logique

Réutilisation des services

Autonomie des services

Optimisation des services

SOAP

Simple Object Access Protocol

- C'est un protocole à part entière, qui va utiliser le support d'autres protocoles (HTTP, SMTP)

Composé

- D'une enveloppe
  - En-tête optionelle
  - Corps du message
- Un modèle de données (format du message)

Initialement défini par Microsoft et IBM

Maintenant une recommandation W3C

SOAP

Utilisé via HTTP (pour faciliter la communication), mais pas seulement

Assez ouvert

Indépendant de la plateforme et du langage

Extensible

SOAP

Basé uniquement sur le format XML

Le client et le serveur communiquent via une « notice » : WSDL

- Web Service Description Language
- Document XML qui décrit
  - Toutes les fonctionnalités (méthodes) disponibles
  - Toutes les structures de données échangeables (objets complexes)

SOAP – AVANTAGES

Haut niveau de sécurité avec WS-Security

Standardisé

Auto-Documenté grâce au WSDL (**W**eb **S**ervice **D**escription **L**anguage)

Extensible

SOAP – INCONVÉNIENTS

Uniquement en XML : lourdeur XML

- Implique un manque de performance

Complexe

Manque de flexibilité

- Couplage reste fort entre clients et serveur (description de la communication WSDL)
- La modification du service implique souvent une évolution côté client

SOAP

WSDL (**W**eb **S**ervice **D**escription **L**anguage) pour la description :

- Services web
- Opérations
- Messages utilisés
- Type des données utilisées
- Protocoles utilisées et leur localisation (URI / URL)

Annuaires UDDI (**U**niversal **D**escription **D**iscovery and **I**ntegration)

SOAP – WSDL

```
<wsdl:types>
  <schema elementFormDefault="qualified"
      targetNamespace="http://soap.webservices.ascadis.fr"
      xmlns="http://www.w3.org/2001/XMLSchema">

    <element name="me" type="xsd:string" />
    <element name="helloYouReturn" type="xsd:string" />
  </schema>
</wsdl:types>
```

Définition des types

SOAP – WSDL

```
<wsdl:message name="helloYouResponse">
  <wsdl:part element="impl:helloYouReturn" name="helloYouReturn" />
</wsdl:message>

<wsdl:message name="helloYouRequest">
  <wsdl:part element="impl:me" name="me" />
</wsdl:message>
```

Définition des messages

SOAP – WSDL

```
<wsdl:portType name="WebService1">
  <wsdl:operation name="helloYou" parameterOrder="me">
    <wsdl:input message="impl:helloYouRequest" name="helloYouRequest" />
    <wsdl:output message="impl:helloYouResponse" name="helloYouResponse" />
  </wsdl:operation>
</wsdl:portType>
```

Définition des opérations (méthodes / fonctionnalités)

SOAP – WSDL

```
<wsdl:binding name="WebService1SoapBinding" type="impl:WebService1">
  <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="helloYou">
    <wsdlsoap:operation soapAction="" />

    <wsdl:input name="helloYouRequest">
      <wsdlsoap:body use="literal" />
    </wsdl:input>

    <wsdl:output name="helloYouResponse">
      <wsdlsoap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

SOAP – WSDL

```
<wsdl:service name="WebService1Service">
  <wsdl:port binding="impl:WebService1SoapBinding" name="WebService1">
    <wsdlsoap:address location="http://localhost:8080/webservices/services/WebService1" />
  </wsdl:port>
</wsdl:service>
```

Définition du service

SOAP – REQUÊTE

```xml
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

  <soap:Body>
    <HelloYouRequest xmlns="http://soap.webservices.ascadis.fr">
      <me>Jeremy</me>
    </HelloYouRequest>
  </soap:Body>
</soap:Envelope>
```

SOAP – RÉPONSE

```xml
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

  <soap:Body>
    <HelloYouResponse xmlns="http://soap.webservices.ascadis.fr">
      <HelloYouReturn>Hello Jeremy</HelloYouReturn>
    </HelloYouResponse>
  </soap:Body>
</soap:Envelope>
```

REST

REpresentational State Transfert

Pas un protocole (tel que HTTP), ni un format

Client-Serveur basé sur le protocole HTTP

Sans état (pas de contexte conservé)

Mise en cache possible et recommandé

Formats divers : XML, JSON, HTML, Texte plein, etc.

REST

Interface uniforme

- Identification des ressources
- Manipulation des ressources

Utilisation des *verbs* HTTP

- GET, POST, PUT, PATCH, DELETE

REST – AVANTAGES

Scalable

Meilleures performances

Facilement manipulable

Flexible

SOAP – REQUÊTE

```
{
  me: "Jérémy"
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<me>Jeremy</me>
```

SOAP – RÉPONSE

```
{
  result: "Hello Jérémy"
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<Result>Hello Jeremy</Result>
```

SERVICES WEB

	SOAP	REST
Définition	Simple Object Access Protocol	REpresentational State Transfer
Design	Protocole standardisé	Style architectural avec recommandations
Approche	Dirigé par les fonctionnalités	Dirigé par les données
Etat	Sans état par défaut, mais possible de le configurer avec état	Sans état
Cache	Cache des appels impossible	Cache des appels possibles
Sécurité	WS-Security avec SSL	HTTPS et SSL
Format	XML	XML, JSON, Texte, YAML, HTML, …
Protocoles de transfert	HTTP	HTTP

SERVICES WEB – NIVEAUX

Niveau 0 – correspond au SOAP
- Toutes les requêtes sont envoyées à la même URL, en POST
- Quelque soit le retour (erreur ou données), status HTTP 200
- Contenu majoritaire en XML

Niveau 1
- Les requêtes sont envoyées à des URL différentes (notion de ressources)

Niveau 2 – correspond au REST
- Utilisation des *verbs* HTTP (GET, POST, PUT, PATCH, DELETE)
- Utilisation des status HTTP (selon erreur ou données – codes de retour)

Niveau 3 – correspond au HATEOAS
- Ajout de liens permettant de naviguer de ressource en ressource
- *RESTful* car la navigation apporte des informations de navigation
  - La documentation n'est pas « nécessaire »

HATEOAS

Hypertext As The Engine Of Application State

Permet, à partir d'une ressource données, de naviguer vers d'autres ressources

- Qui sont en lien avec la ressource consultée
  - Depuis un produit, aller vers son fournisseur, puis du fournisseur, aller vers la liste des produits

Attention, ce n'est pas un standard !

- Quelques expérimentations : *HAL* ou *JSON API*

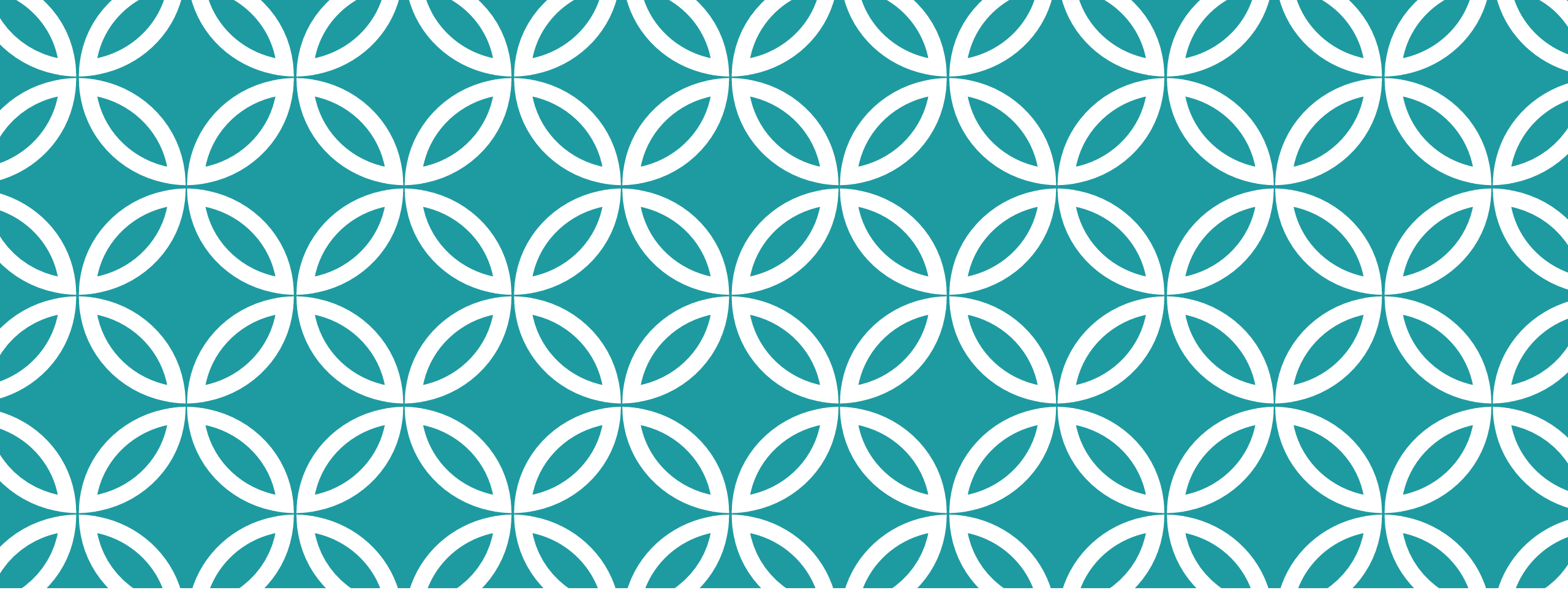
HATEOAS


```
{
  "libelle": "Voile principale Safire",
  "prix": 2199.99,
  "_links": {
    "self": {
      "href": "http://www.ascadis.fr/api/produits/1"
    },

    "achats": {
      "href": "http://www.ascadis.fr/api/produits/1/achats"
    },

    "fournisseur": {
      "href": "http://www.ascadis.fr/api/produits/1/fournisseur"
    },

    "list": {
      "href": "http://www.ascadis.fr/api/produits{?page,size,sort}"
    }
  }
}
```



TRAVAUX PRATIQUES

TRAVAUX PRATIQUES

Calculatrice (REST + SOAP)

- Additionner(a, b)
- Soustraire(a, b)

Mettre en place **SWAGGER** pour l'API REST

Créer un client REST et un client SOAP