Architecture Orientée Services

Chapitre 1 : Introduction

Année universitaire : 2024-2025

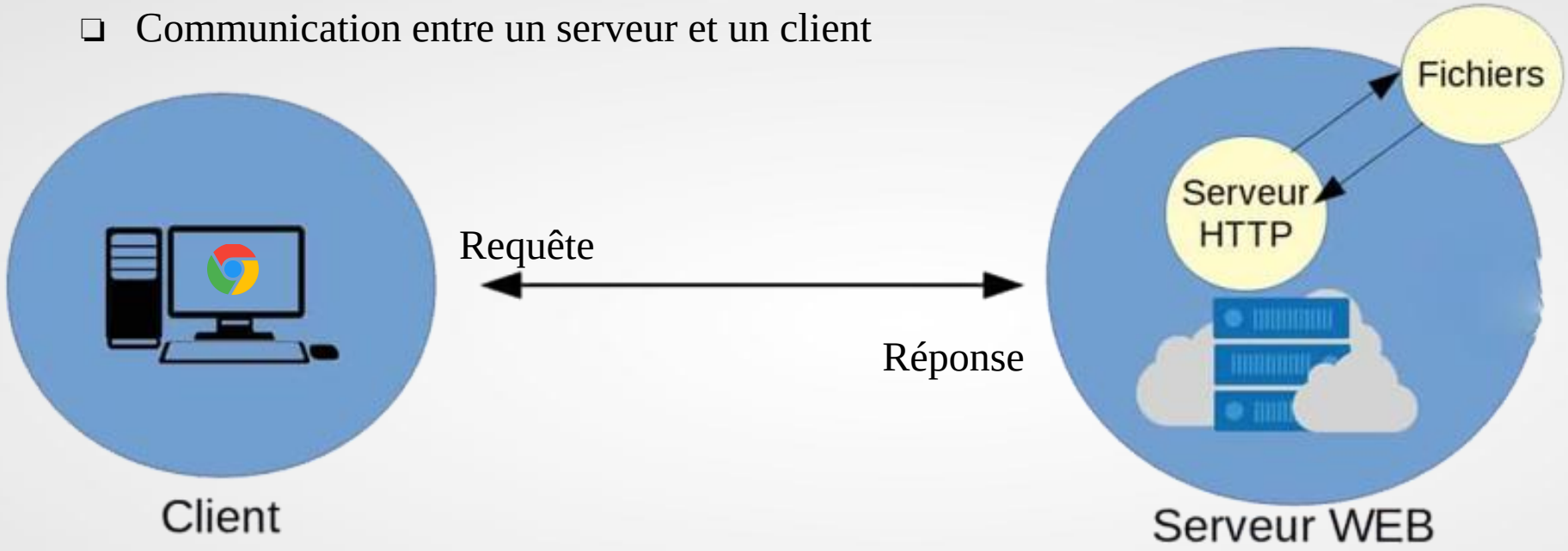
Plan

- Le Fonctionnement Du Web
- Les Standards liés au web
- Les paradigmes Logiciels
- Les techniques de communication sur le web
- L’architecture SOA

Le Fonctionnement du web

Fonctionnement du Web

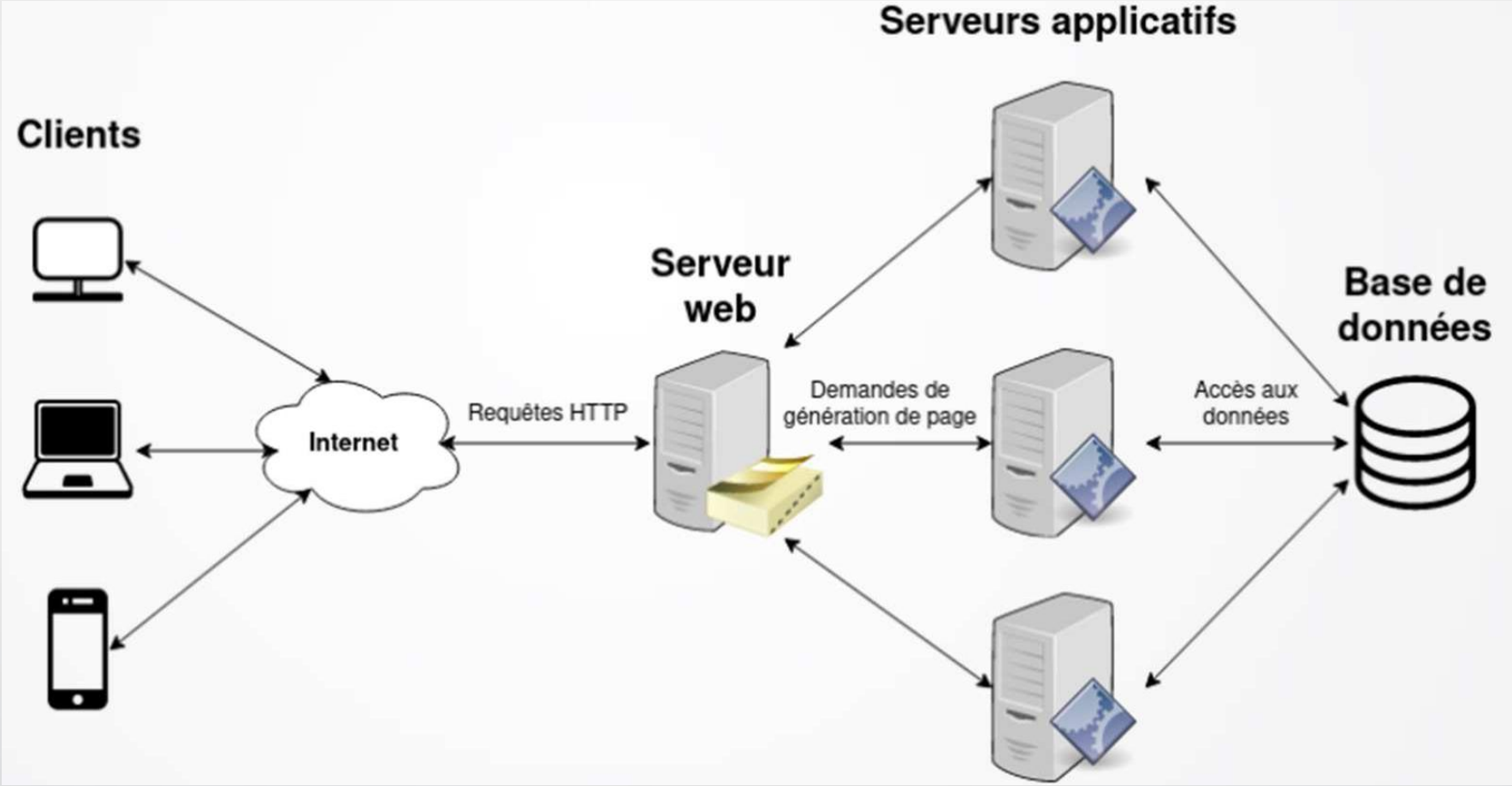
- Communication entre un serveur et un client



- Le Web est désormais indispensable pour accéder à l'information, établir des communications et interagir avec divers services en ligne à l'échelle mondiale.

Systèmes basés sur le Web

- Fonctionnement



Systèmes basés sur le Web

❑ Couches

- **Client (navigateur)**: affiche les informations à l'utilisateur
- **Serveur Web**: gère les requêtes HTTP entrantes en traitant les données statiques et retourne le résultat finale au Client
- **Application**: exécute des applications web dynamiques en traitant la logique métier et en communiquant avec d'autres services
- **Données**: stocke les données utilisées par le serveur applicatif

Standards liées au Web

Les Standards Fondamentaux du Web : Formats, Protocoles et Échanges

- Les standards liées au Web comprennent des formats de données tels que HTML, XML et JSON pour structurer et échanger des informations.
- Le transfert de ces données est réalisé via les protocoles HTTP et TCP, utilisant des URI/URL pour localiser les ressources.
- Les requêtes et réponses HTTP sont organisées selon une structure spécifique, et les méthodes HTTP permettent d'effectuer différentes actions sur les ressources du Web.

Formats d'échange de données sur le Web

- **TEXT**
- **HTML (HyperText Markup Language)**: Langage de base du Web pour la création de pages web.
- **XML (eXtensible Markup Language):** Langage de balisage utilisé principalement pour la structuration et l'échange de données.
- **JSON (JavaScript Object N**Format léger et lisible pour l'échange de données structurées. Utilise des paires "clé : valeur".**otation):**

Protocole de communication TCP vs HTTP

- **TCP** (Transmission Control Protocol) est un protocole de la couche de transport qui assure une communication fiable et orientée connexion entre les appareils sur un réseau.

- **HTTP** (Hypertext Transfer Protocol) est un protocole de la couche application qui s'exécute au-dessus de TCP (ou d'autres protocoles de la couche de transport) et régit le transfert de ressources hypertexte (web) entre les clients et les serveurs sur le World Wide Web.

- HTTP s'appuie sur TCP pour assurer un transport fiable des données entre les navigateurs et les serveurs web

HTTP: Méthodes

- Les méthodes HTTP spécifient l'action à effectuer sur la ressource:

  • GET – Obtenir des informations à partir d'un serveur Web.

  • POST – Soumettre des données au serveur Web et potentiellement créer de nouveaux enregistrements.

  • PUT – Mettre à jour une ressource existante avec de nouvelles données.

  • PATCH – Mettre à jour une partie d'une ressource existante, sans modifier l'intégralité de la ressource.

  • DELETE – Supprimer des informations/enregistrements d'un serveur Web.

HTTP: Méthodes

• HEAD — Récupérer les en-têtes de réponse d'une ressource, sans le corps de la réponse.

- Souvent pour vérifier si une ressource existe ou a été modifiée depuis la dernière demande.

- Exemple: vérifier si une image est toujours disponible sur un serveur externe.

• OPTIONS — Découvrir les méthodes HTTP prises en charge par une ressource.

- Cela permet au client d'adapter ses requêtes ultérieures en fonction des actions possibles sur la ressource.

- Exemple: Un client qui rencontre une ressource pour la première fois.

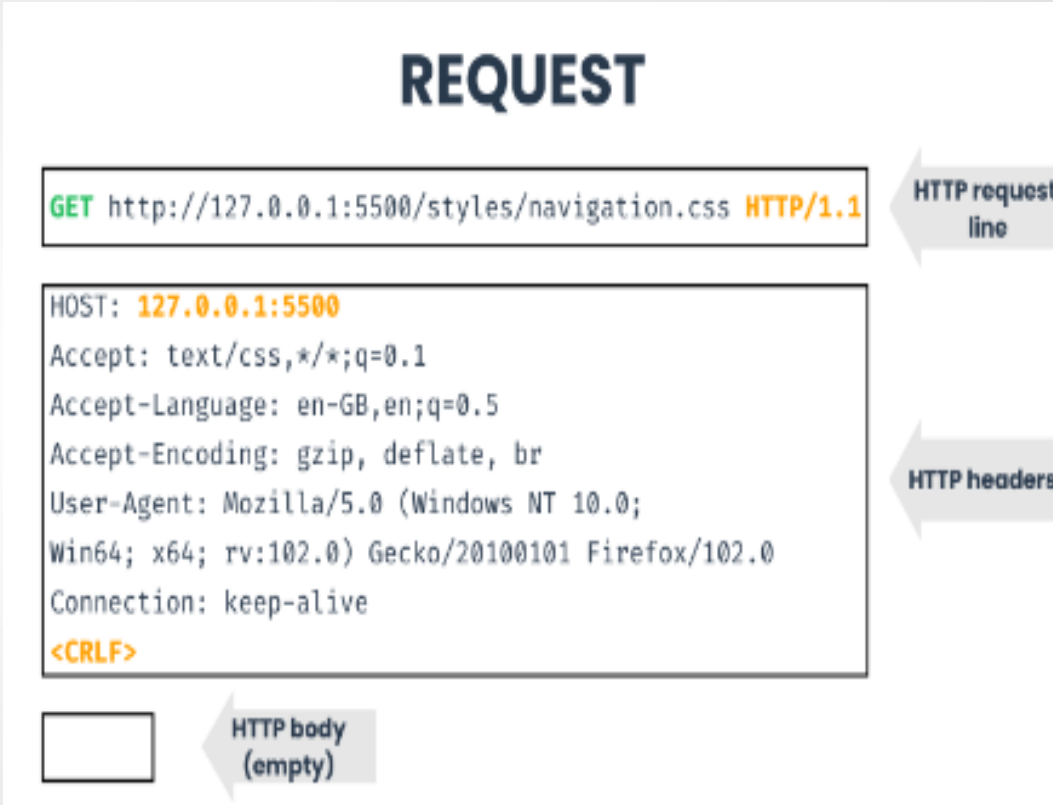
HTTP: Structure de Requête

La requête HTTP est composée de trois parties principales :

• Ligne de Requête – Indique la méthode HTTP, l'URL de la ressource demandée et la version du protocole HTTP.

•En-têtes de Requête – Fournissent des informations supplémentaires sur la requête, telles que le type de contenu, les cookies, etc.

•Corps de Requête (facultatif) – Contient des données envoyées au serveur généralement utilisé avec les méthodes POST, PUT et PATCH).



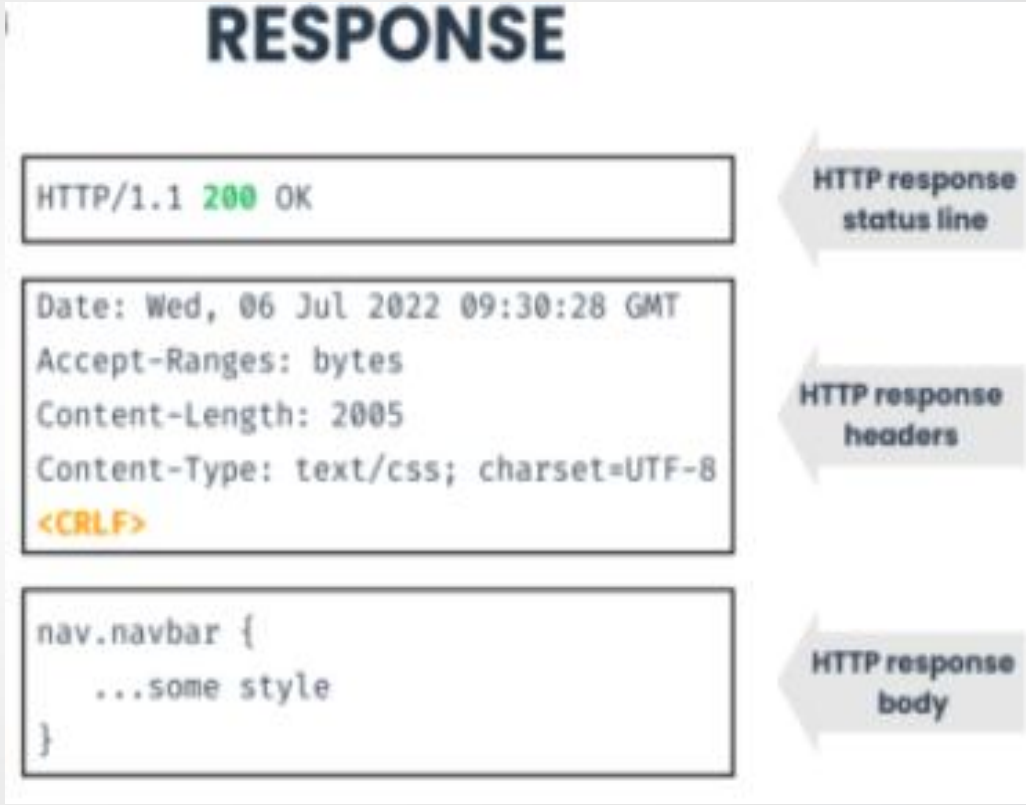
HTTP: Structure de Réponse

La réponse HTTP est aussi composée de trois parties principales :

• Ligne de Statut – Indique la version du protocole HTTP, le code du statut HTTP et sa description.

•En-têtes de Réponse – Fournissent des informations supplémentaires sur la réponse, telles que le type de contenu, la date de dernière modification et les cookies.

•Corps de Réponse (facultatif) – Contient le contenu demandé par le client, généralement du HTML, des images, des vidéos ou des données JSON.



HTTP: Structure de Réponse

Les codes de statut HTTP indiquent le résultat de la requête. Ils sont divisés en cinq catégories :

1xx (Informationnel)

La requête a été reçue et le processus est en cours.

Exemple: 100 Continue

2xx (Succès)

La requête a été reçue, comprise et acceptée.

Exemple: 200 OK | 201 Created | 204 No Content

3xx (Redirection)

Des actions supplémentaires doivent être prises par le client pour terminer la requête.

Exemple: 302 Found | 304 Not Modified

4xx (Erreur Client)

La requête contient une mauvaise syntaxe ou ne peut être s satisfaite.

Exemple: 400 Bad Request | 401 Unauthorized | 404 Not Found

5xx (Erreur Serveur)

Le serveur a échoué à satisfaire une requête valide.

Exemple: 500 Internal Server Error

URL vs URI

Tous les URL sont des URI, mais tous les URI ne sont pas des URL.

❑ **URI (Uniform Resource Identifier) :**

- Offre un moyen général d'identifier des ressources sur le Web ou dans un réseau.

- Il peut s'agir d'une page web, d'une image, d'une vidéo, d'un fichier ou de tout autre type de ressource accessible via un protocole réseau.

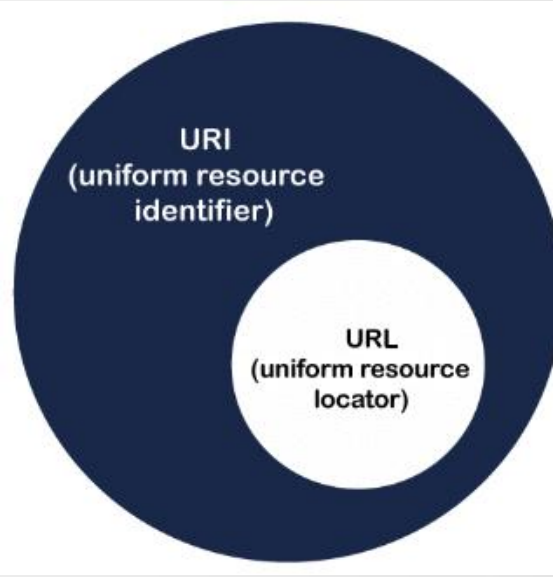
- Peut utiliser différents protocoles (HTTP, FTP, HTTPS, etc).

❑ **URL (Uniform Resource Locator) :**

- Un type spécifique d'URI.

- Spécifiquement conçue pour localiser des ressources web via HTTP.

- S'agit de l'adresse la plus courante utilisée pour accéder aux pages web sur Internet.

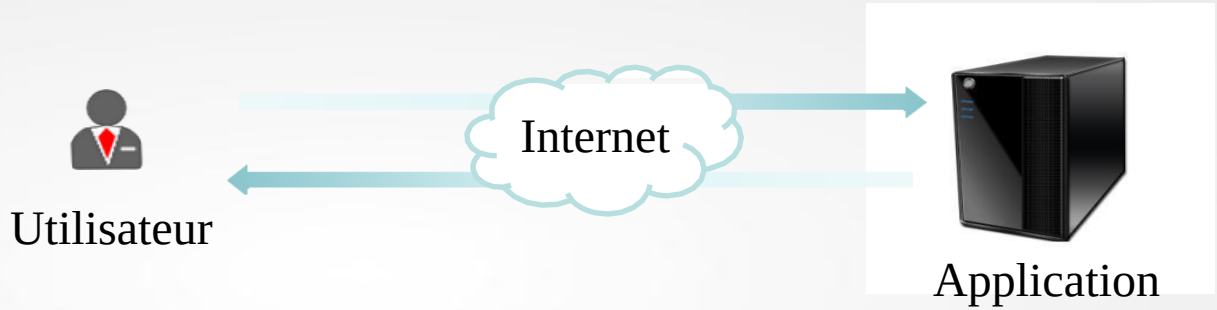


Contexte de communication

Human-centric web

- Le Web centré utilisateur implique que l'humain est l'acteur principal pour l'initialisation de l'ensemble des requêtes

B2C
Business ToConsumer



Application-centric web

- Le Web centré application a pour objectif de permettre à des organisations de communiquer entre elles

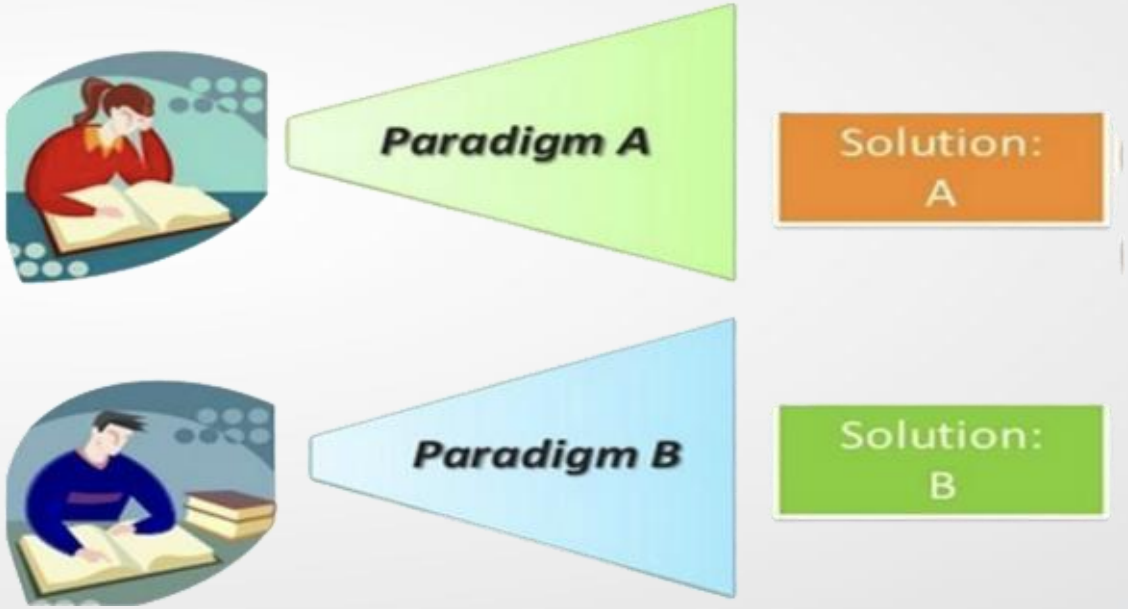
B2B
Business To Business

Les paradigmes logiciels

Paradigmes des architectures logicielles

Le terme de paradigme est employé pour exprimer la façon dont un système a été conçu et pensé dans ses grandes lignes.

Objectif:
Développer une application de facturation



Paradigmes des architectures logicielles

- Les révolutions informatiques coïncident généralement avec un changement de paradigme.
- Niveau d'abstraction grandissant avec l'évolution des paradigmes

- On distingue l'évolution des paradigmes logicielles comme suit:

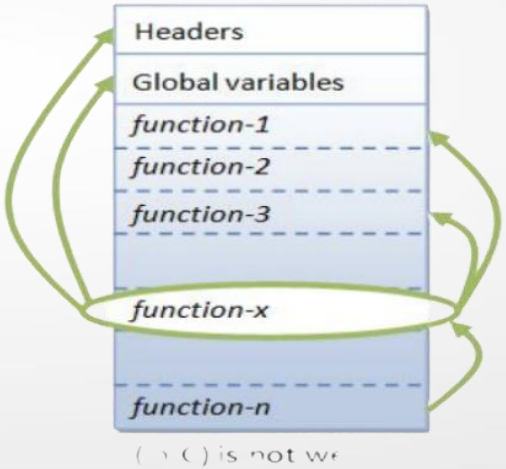
Paradigmes des architectures logicielles

Paradigme procédural

- ❏ le code est organisé en procédures (ou fonctions) qui contiennent des séquences d'instructions. Ces procédures sont appelées successivement pour accomplir une tâche spécifique.
- ❏ Un Système informatique **désordonné**

Limites :

- ❏ Tend à générer ducode "Spaghetti"
- ❏ Maintenance complexe
- ❏ Modularité et abstraction absente
- ❏ Réutilisation ardue



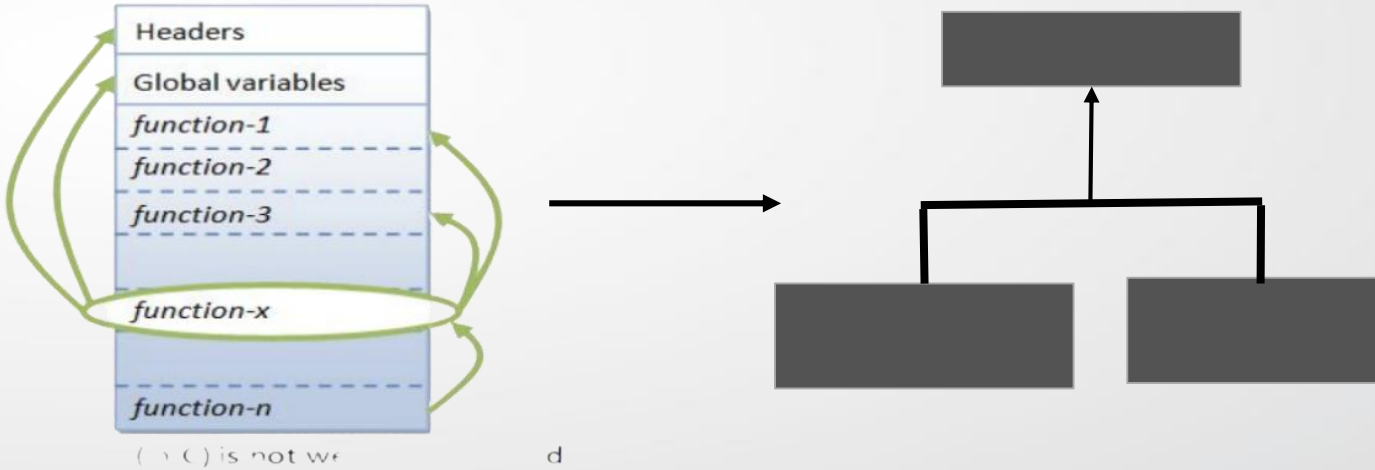
Paradigmes des architectures logicielles

Paradigme objet

- L'idée est de concevoir les programmes non plus comme des lignes de codes qui s'exécutent séquentiellement, mais comme des objets qui dialoguent
- Ses principes incluent l'abstraction encapsulation,de données, polymorphisme et héritage.

Limites :

- Réutilisation difficile
- Couplage fort
- Problème de maintenance



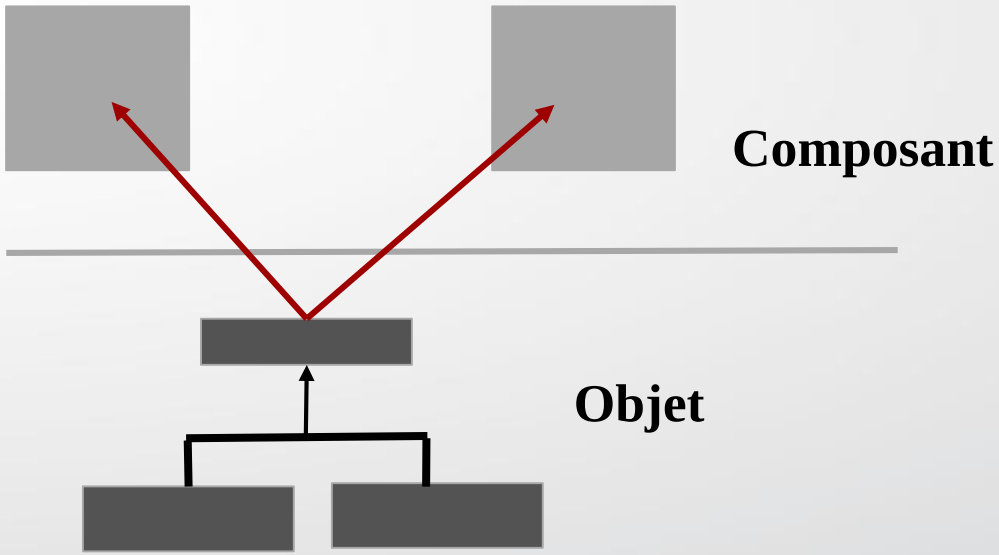
Paradigmes des architectures logicielles

Paradigme composant

- Construire une application composée par un ensemble de briques de base configurables et réutilisables.
- Il s'agit d'externaliser le code fonctionnel d'une application afin de le rendre réutilisable dans d'autres applications

Limites :

- Complexité de gestion des dépendances
- Interopérabilité entre composants hétérogènes



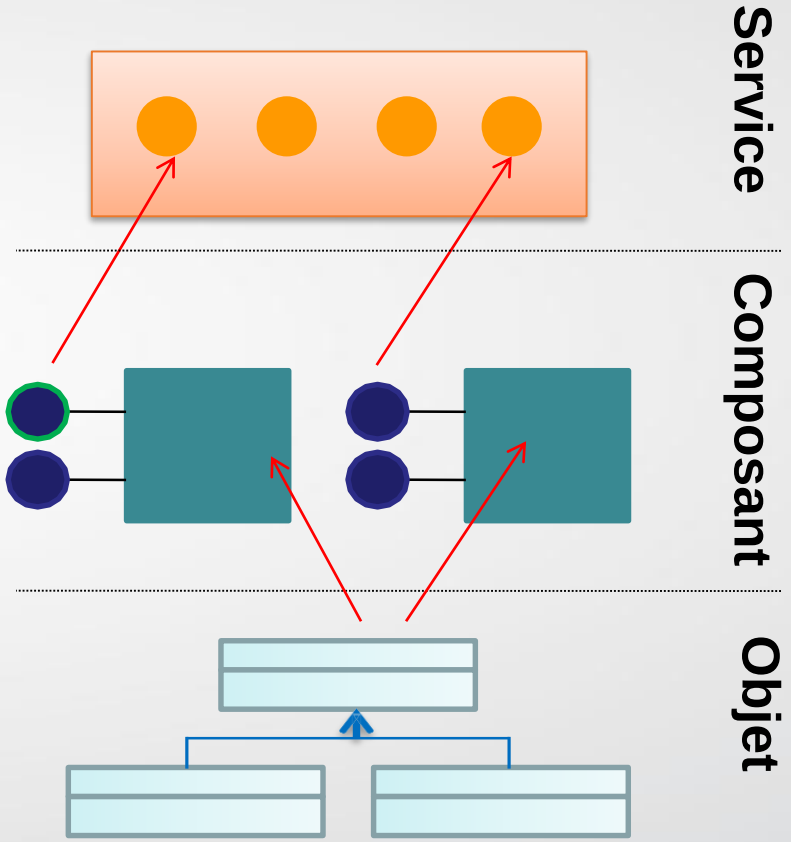
Paradigmes des architectures logicielles

Paradigme service

- Prise en charge de la diversité et de l'hétérogénéité des termes de langages systèmes de programmation, et de logiciels, en de technologies de conception (et de réalisation) ou de plates-formes d'exécution

- Le paradigme service permet de:

- réduire le couplage
- améliorer la réutilisation
- augmenter l'abstraction

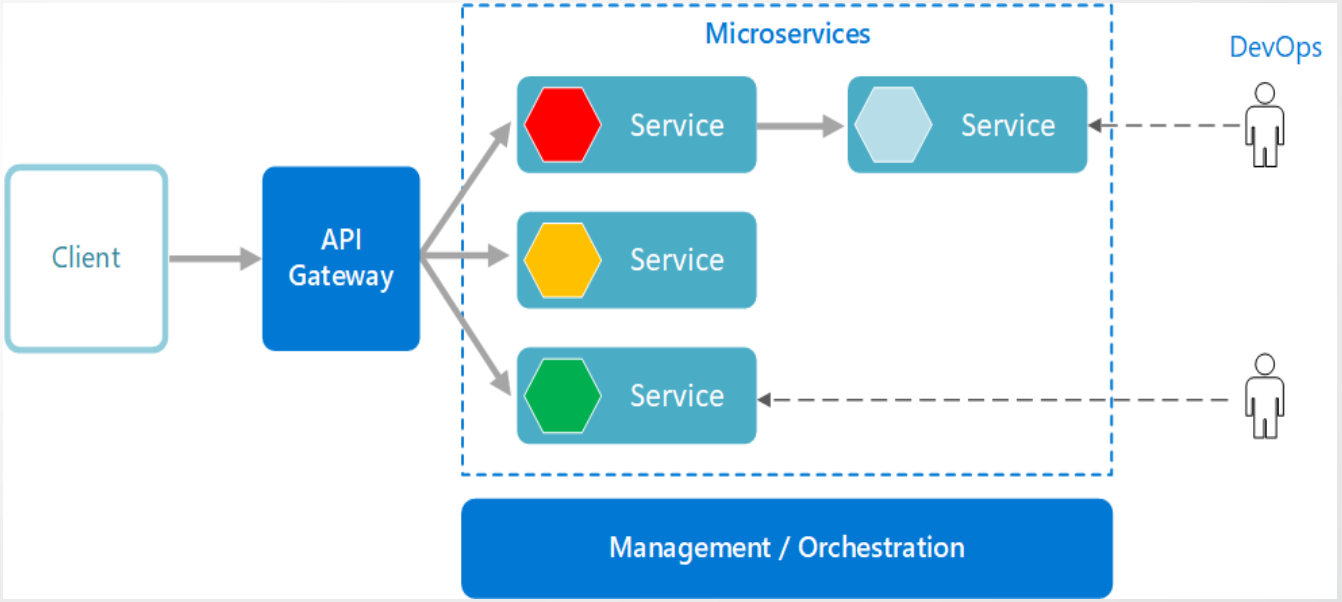


Paradigmes des architectures logicielles

Paradigme Microservice

- Prise en charge de la décentralisation et de l'autonomie des composants logiciels, tout en permettant une diversité technologique (langages, frameworks, bases de données) et une indépendance des cycles de déploiement.
- Le paradigme Microservice permet de:

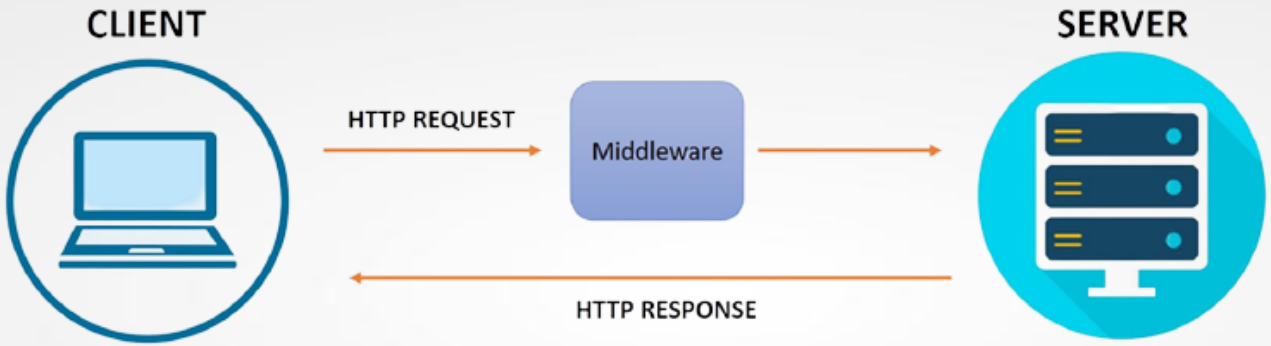
- Réduire le couplage
- Améliorer la scalabilité
- Faciliter l'évolution et la maintenance
- Garantir une hétérogénéité technologique, chaque service pouvant utiliser le stack technique la plus adaptée à son domaine.



Les techniques de communication sur le web

Middlewares

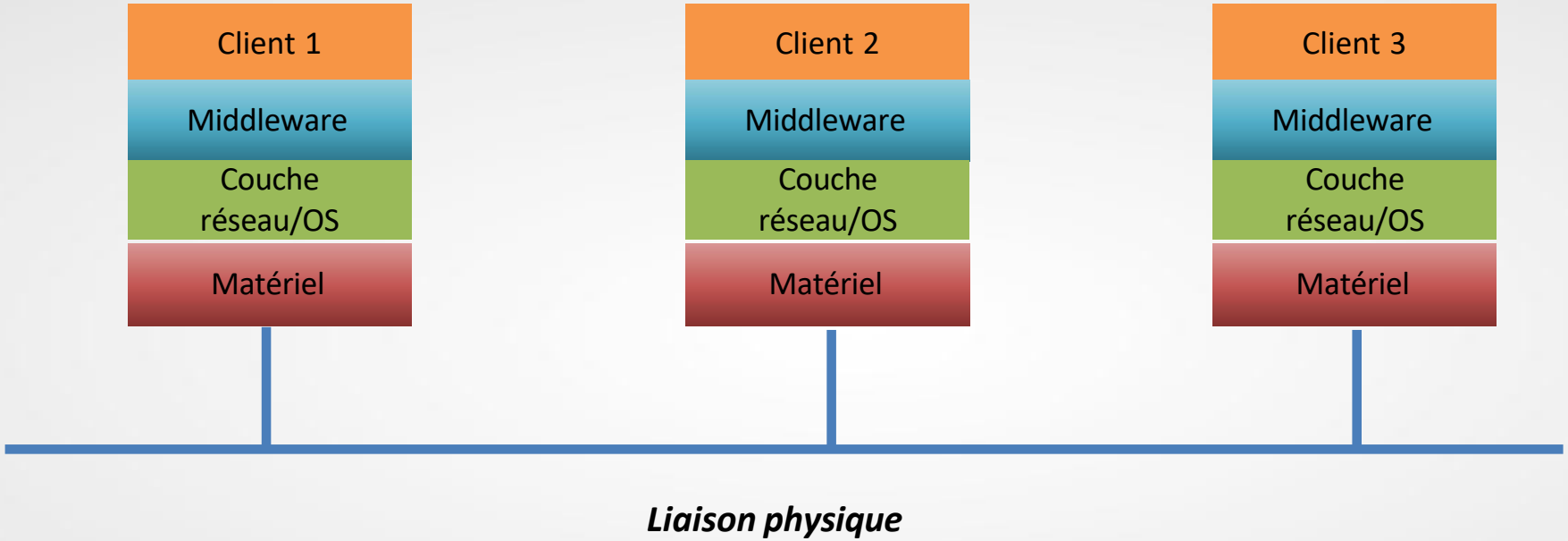
Un logiciel servant d'intermédiaire entre d'autres logiciels
Un intermédiaire de communication entre des applications complexes et distribuées



Le middleware joue un rôle essentiel en simplifiant la communication et l'intégration entre applications, services ou composants logiciels dans un environnement distribué.
Il agit comme une couche d'abstraction, permettant aux différents éléments du système de travailler harmonieusement ensemble, indépendamment de leurs différences techniques, langages de programmation ou plates-formes de développement.

Middlewares

Fonctionnement:



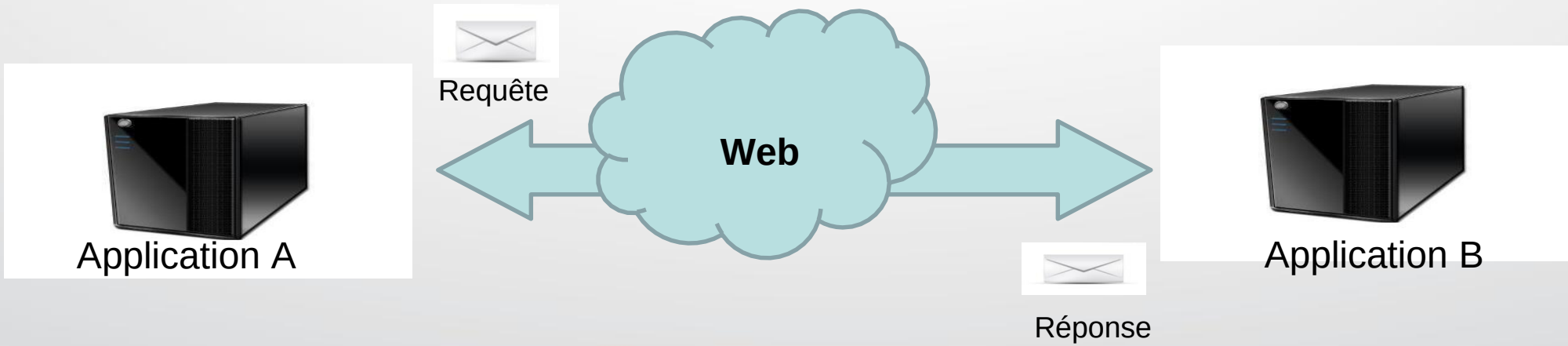
Rôles de base:

- **Résoudre l'intéropérabilité : Unifier l'accès à des machines distantes**
- **Résoudre l'hétérogénéité : Etre indépendant des systèmes d'exploitation et du langage de programmatio des applications**

28

Middlewares

- Solutions existantes:
  - DCOM,
  - .NET Remoting
    - Middlware .Net
  - RMI (Remote Method Invocation)
    - Middlware Java permet de faire communiquer des *objets java* **distribués** sur le réseau
  - CORBA (Common Object Request Broker Architecture)
    - Permet de faire communiquer des objets écrits dans des langages différents (C++, Java, Smalltalk)
  - **Quels sont les atouts d'une meilleure solution?**

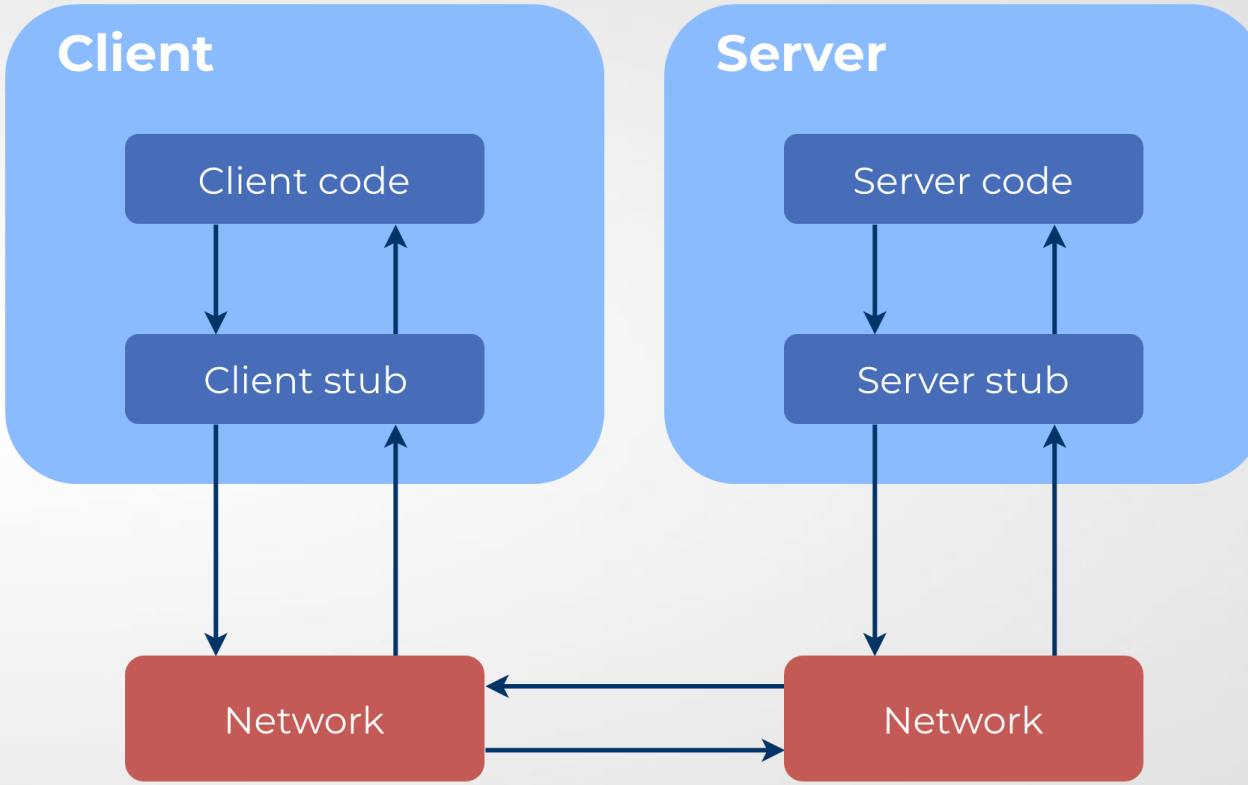


Middlewares: RPC

RPC (Remote Procedure Call)

Se composent de trois composants principaux :

- Un client (l'appelant): le composant qui effectue l'appel à distance;
- Un serveur (l'appelé): le composant qui implémente la procédure invoquée;
- Un langage de définition d'interface (IDL): le langage par lequel le client et le serveur communiquent.



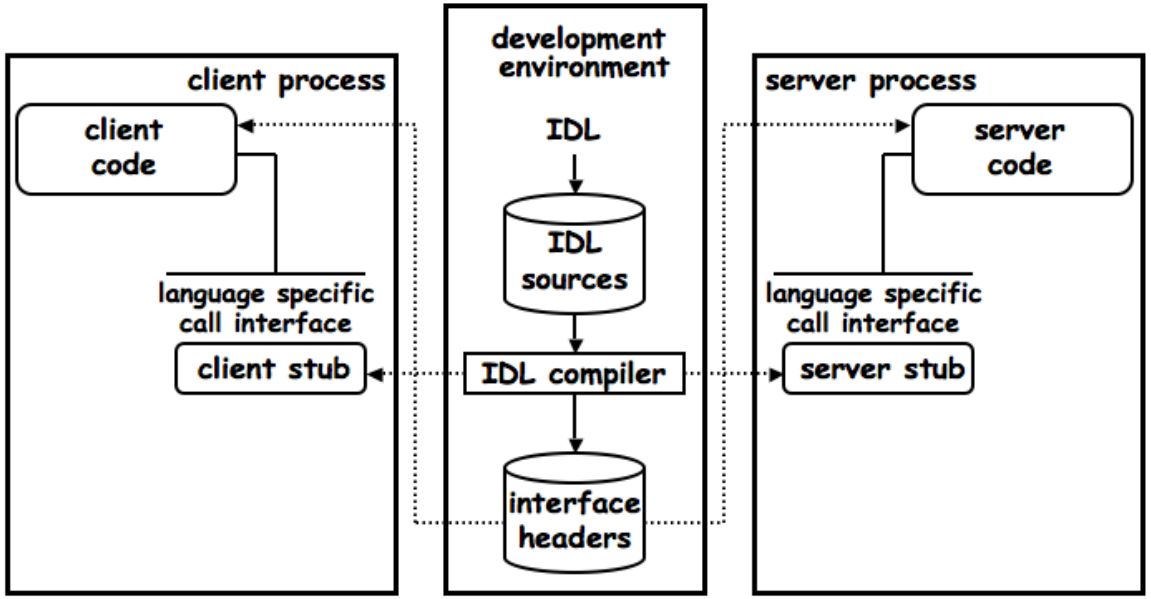
Architecture Middelware RPC [1]

Middlewares: L'IDL

IDL (Interface Definition Language)

IDL: le premier composant implémenté

- Utilisé pour définir les procédures du serveur qui sont disponibles pour le client.
- Décrit les paramètres d'entrée ainsi que la réponse renvoyée.
- Indique au client quels services distants sont disponibles
- Comment ils sont accessibles et quelle sera la réponse du serveur



Interface Definition Langage [10]

Middlewares: CORBA

CORBA (Common Object Request Broker Architecture)

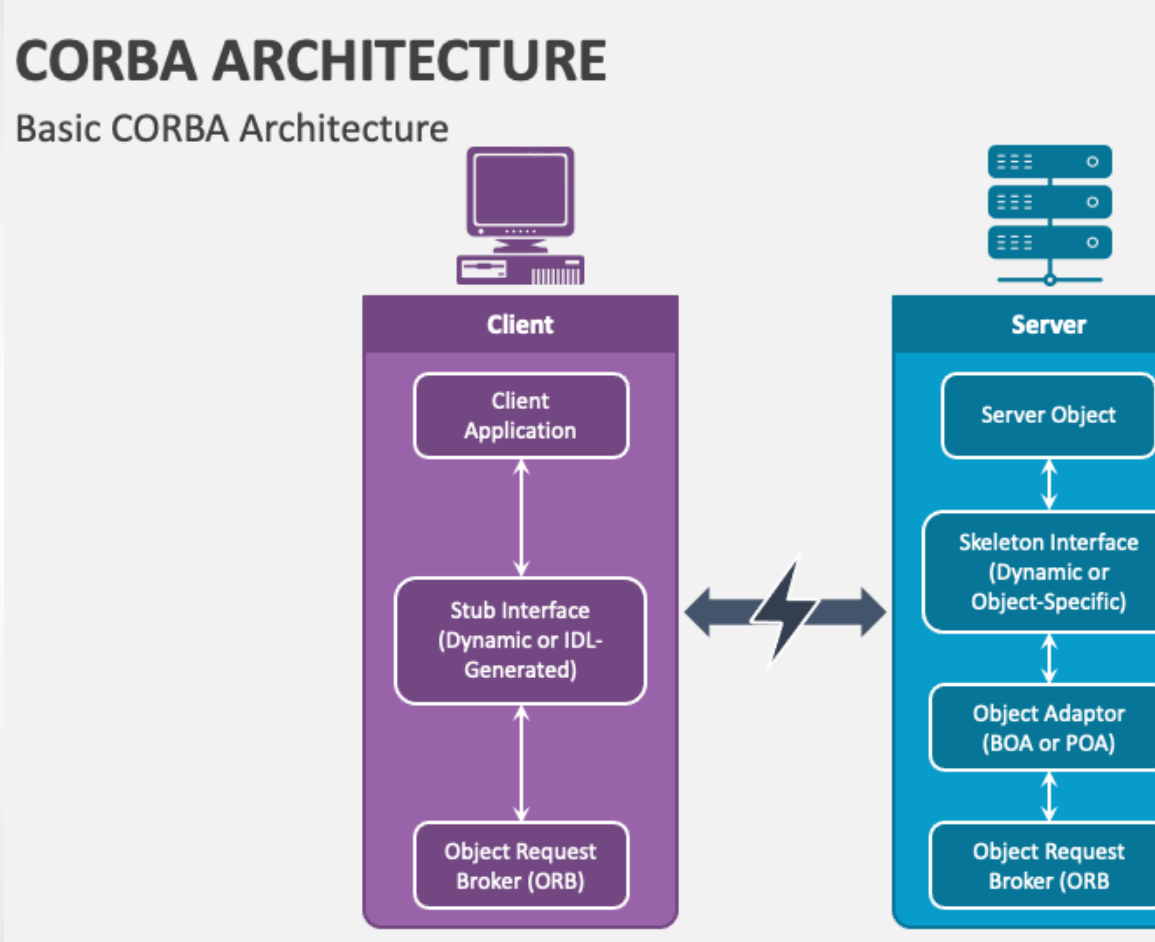
Se compose de trois composants principaux :

•**Un client (l'appelant)** : demande un service en invoquant une méthode sur un objet distant.

•**Un serveur (l'appelé)** : implémente l'objet distant et fournit les services demandés par le client.

•**Un langage de définition d'interface (IDL)** : utilisé pour définir les interfaces des objets accessibles à distance, permettant l'interopérabilité entre différents langages de programmation.

. ORB (Object Request Broker): C'est le cœur de CORBA : il assure la communication entre clients et objets distants. Il localise l'objet, transmet l'appel, et retourne la réponse.



Architecture Middelware CORBA [2]

Middlewares: CORBA VS RPC

Critère	RPC (Remote Procedure Call)	CORBA (Common Object Request Broker Architecture)
Langage d'interface (IDL)	Parfois spécifique à l'implémentation (ex. Sun RPC IDL)	Utilise un **IDL standardisé** (CORBA IDL) indépendant des langages
Interopérabilité	Limitée, souvent dépendante de la plateforme	**Forte interopérabilité** entre langages et plateformes
Complexité	Plus **simple** à implémenter	Plus **complexe** mais plus riche en fonctionnalités
Support des objets distribués	Non (seulement des fonctions)	Oui (objets distants, héritage, polymorphisme)
Langages pris en charge	Dépend de l'implémentation (souvent C/C++)	Multi-langage (Java, C++, Python, Ada, etc.) grâce à l'IDL
Exemple typique	Sun RPC, ONC RPC, Java RMI	ORBacus, TAO, JacORB

Le terme Service

Définition

- Un service est un terme général utilisé en informatique pour décrire une unité autonome de fonctionnalités qui peut être exécutée de manière indépendante.
- Il possède une interface pour la communication et l'interaction avec un demandeur de service.

- Les services peuvent être à la fois locaux ou distants :
  - ✓ Locaux: s'exécutent sur la même machine que l'application qui les consomme ou bien en internes à une organisation comme **les services système** ou **d'application de bureau**;

  - ✓ Distants: existent sur le Web et peuvent s'exécuter sur des machines différentes ou être hébergés dans le cloud comme les services web ou **cloud**.

Services Web

- Service web = **service + web**
- Un service Web met à disposition un service **via Internet**

- Un SW est un **programme informatique**, permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués.
- Les services Web interagissent à travers l'échange de messages

Services Web

Il constitue ainsi une interface permettant à deux machines (ou applications) de communiquer. Pour y parvenir, la technologie doit disposer de deux propriétés essentielles :

- **être multiplateforme** : il n'est pas nécessaire que le client et le serveur aient la même configuration pour pouvoir communiquer. Le service Web leur permet de se retrouver à un même niveau.
- **être partagée** : dans la plupart des cas, un service Web est à disposition de plus d'un client. Différents clients accèdent à ce service via Internet.

Service Web VS API

Tout service web est une API (Interface de Programmation d'Applications)
Tout API n’est pas un forcément un service web (l'inverse n'est pas toujours vrai)

- **Une API** est une interface qui permet à différentes applications ou systèmes de communiquer entre eux.
- Les API **ne nécessitent pas forcément d'utiliser le réseau ou Internet**.

- **Un service web** (API web) est un type spécifique d'API qui **utilise des protocoles web** pour permettre la communication entre machines sur un réseau, généralement **Internet**.
- Fonctionnent sur des machines différentes, souvent via Internet ou un réseau interne.

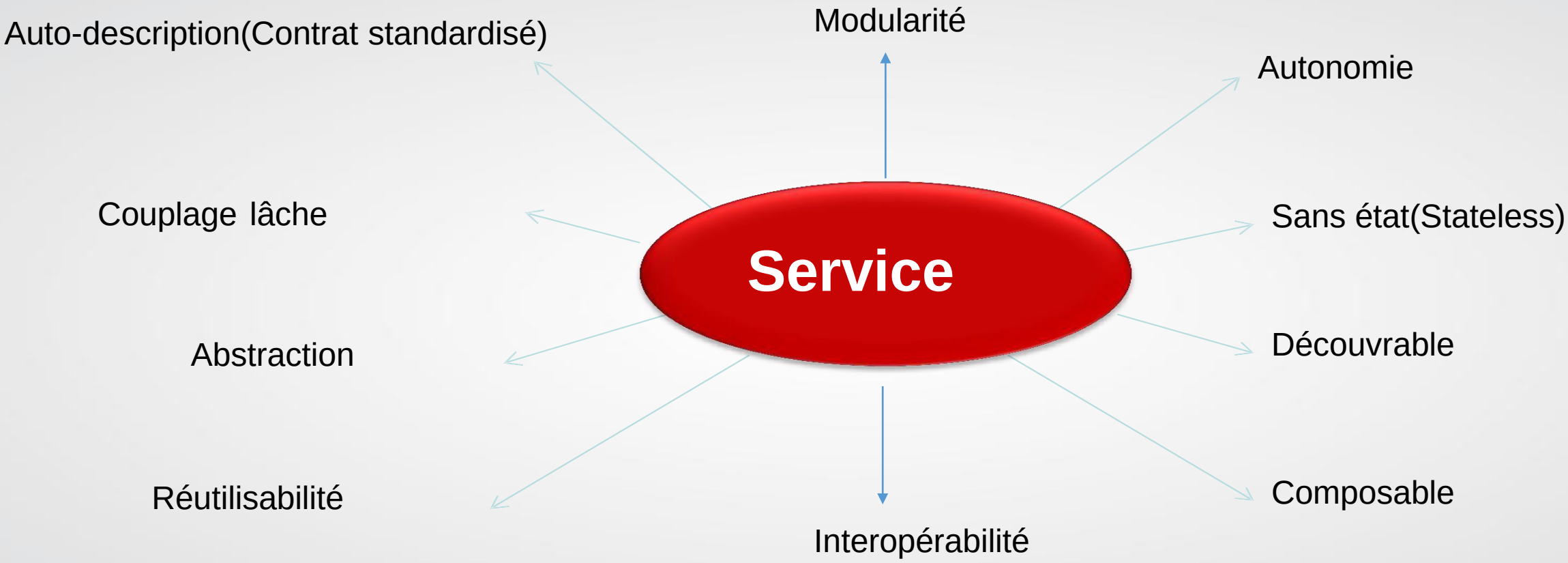
Famille de Services Web

❑Il existe deux grandes familles de services web:

- Les services web **étendus** utilisant les standards:
  - SOAP pour la communication ;
  - UDDI (annuaire) pour la publication ;
  - WSDL (contrat) pour la description

- Les services web **REST** utilisant :
  - Directement HTTP au lieu d'une enveloppe SOAP ;
  - Un URI pour nommer et identifier une ressource ;
  - Les méthodes HTTP (POST, GET, PUT et DELETE) pour effectuer les opérations de base CRUD

Principes d'un service Web



Principes d’un service Web

➢Auto-description (Contrat standardisé)

L’ensemble des services d’un même Système Technique sont exposés au travers de contrats respectant les mêmes règles de standardisation.

➢Abstraction

Le service fonctionne en « **boîte noire** »

- Seul le contrat du service (informations nécessaires pour l’invocation) est exposé au consommateur du service .

- Le fonctionnement interne du service (sa logique métier et son implémentation) n’est **pas visible** .

Principes d’un service Web

Couplage lâche

- Dépendance faible entre le consommateur et le service
  - **Dépendance du contrat** et non pas de l’implémentation;
  - Echange à travers des **messages;**
  - Orchestration qui assure l’indépendance des services vu qu’elle leur permet de communiquer pour réaliser un processus, sans avoir à se connaître

Principes d’un service Web

➢ Autonomie

- Un service ne doit être dépendant d'aucun contexte ou service externe:
  - Son comportement est indépendant du contexte fonctionnel et technique dans lequel il a été invoqué.

➢ Stateless

- Un service ne stocke pas les informations des clients:
  - Ne stocke pas de données;
  - Ne fait référence à aucune transaction passée.

Principes d'un service Web

- Composable

➢ Un service peut participer à des compositions de services

- Un ensemble de services peuvent être composés à travers leur orchestration pour répondre à un besoin complexe;
- **Avantage:**
  - Apport de valeur ajoutée (répondre à un nouveau besoin complexe)

Principes d'un service Web

Modularité

La modularité est matérialisée par le fait que les différents services sont indépendants les uns des autres dans la plateforme de services et peuvent être accessibles par les applications distantes tournant dans des environnements hétérogènes. D'où la notion d'interopérabilité.

Interopérabilité

Réfère à la capacité d'un système à coexister et à coopérer avec d'autres systèmes éventuellement **hétérogènes** selon un schéma ouvert d'interconnexion.

- ✓Langages de programmation différents,
- ✓SGBD différents,
- ✓ Architectures différentes,
- ✓etc.

Principes d’un service Web

- Intégration vs Interopérabilité –

- **Integration** means that you've writtensome custom code to connect two (or more) systems together.
  - *[Bobby Wolf – IBMArchitect]*

- Lorsqu'on parle d'intégration, nous pensons au processus qui fait que différents systèmes d'information **apparaissent comme un seul**.

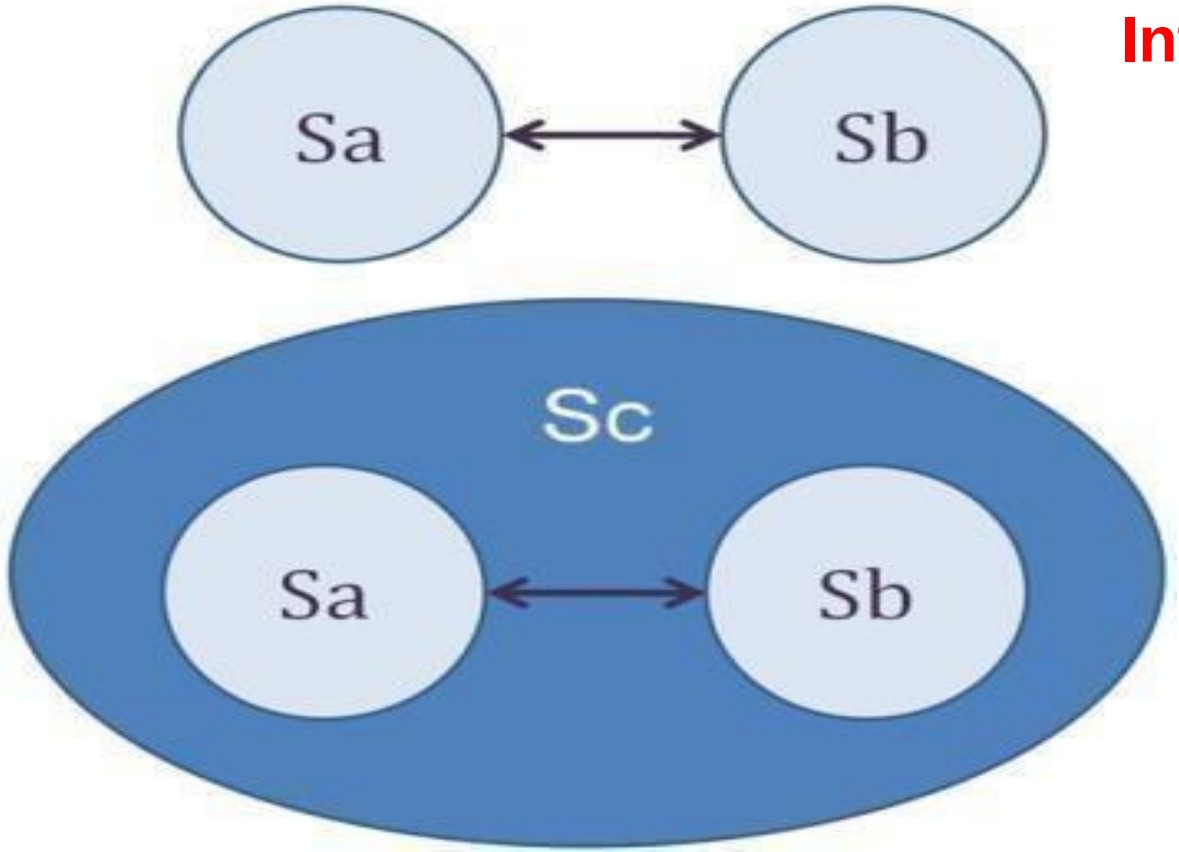
Principes d'un service Web

- Intégration vs Interopérabilité –

Two heterogeneous Systems Sa and Sb

One « integrated » system Sc

(Does not mean that Sa and Sb have been « merged »)



Interoperabilité

Integration

Source: [3]

Avantages des services web

- Offrir une technologie adaptée aux applications B2B;

- Rendre possible et plus facile l'interconnexion et l'interaction des systèmes et composants hétérogènes;
- Utilisés par le Web Sémantique pas seulement le web interactif

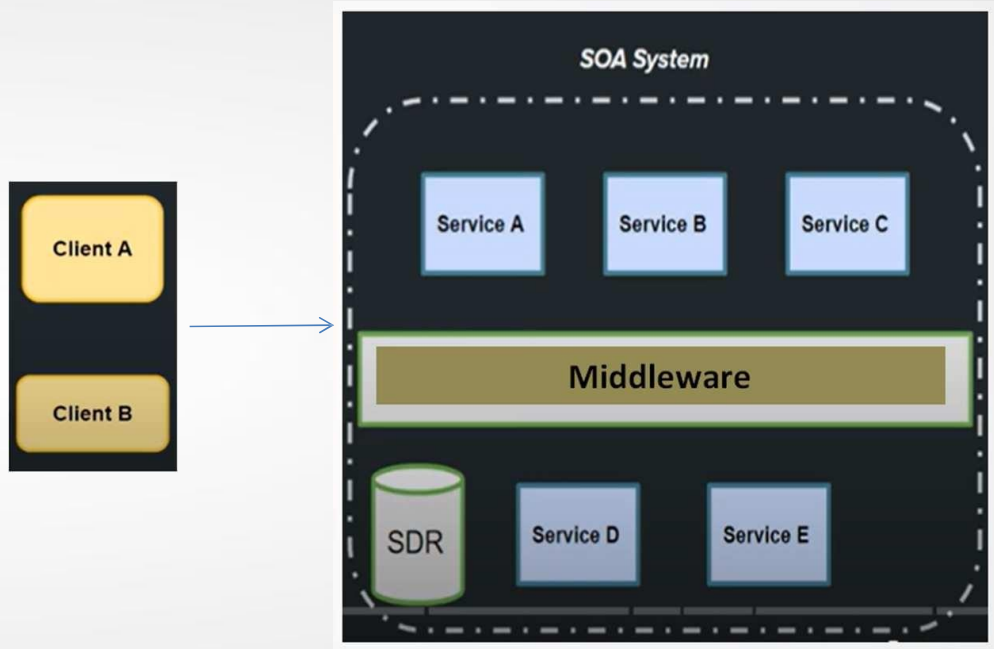
- Garantir l'interopérabilité et donner lieu à des systèmes plus ouverts que ceux utilisant des protocoles tels que RPC, DCOM, RMI… ;
- Réutilisables dans un environnement ouvert;

- Garantir un couplage lâche.

L’architecture SOA

Présentation SOA

- L'architecture orientée services concerne la manière de créer, d'utiliser et de combiner des **services**.



Présentation SOA

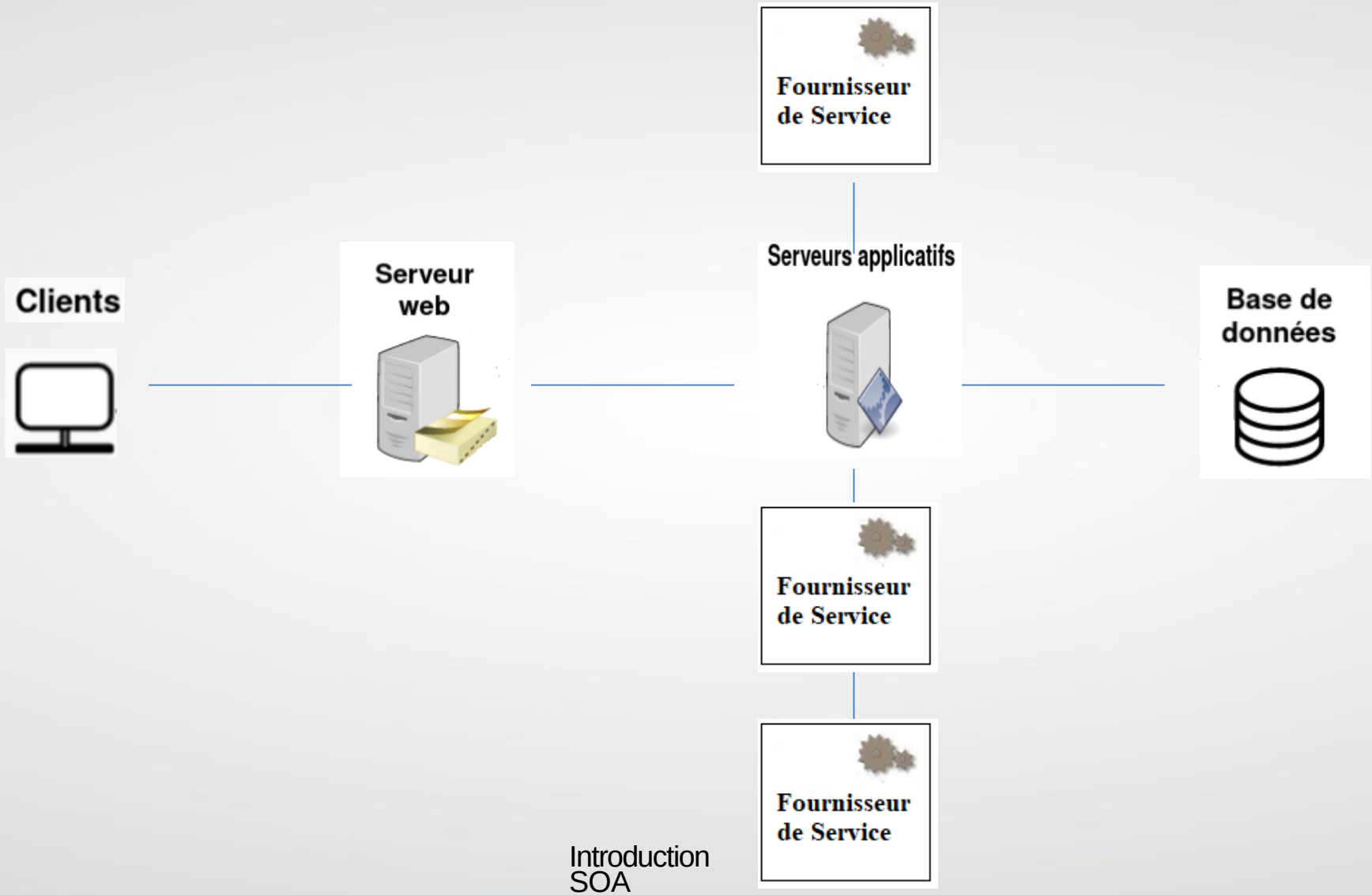
- “ L’architecture orientée service constitue un **style d’architecture** basée sur le principe de séparation de l’activité métier en une série de **service**s”.

“ Ces services peuvent être assemblés et liés entre eux selon le principe de couplage lâche pour exécuter l’application désirée. ”

Gartner - Septembre 2005

- Objectifs: Décomposer une fonctionnalité en un ensemble de fonctions basiques(services) fournies par des composants.
- Décrire finement le schéma d’interaction entre ces services.

Couches de vue de Services



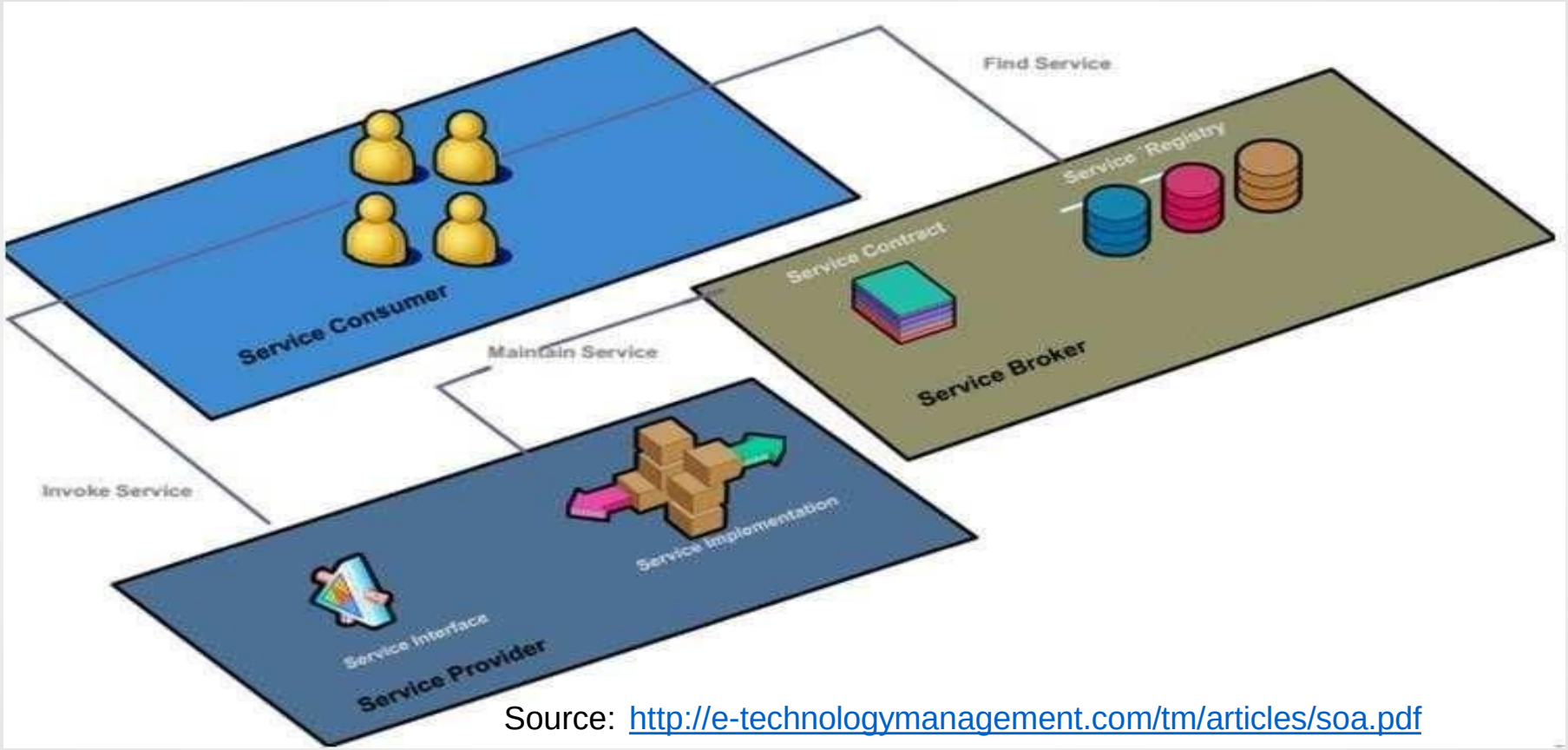
Elèments SOA

- Le fournisseur de service créé le service Web, puis publie son interface ainsi que les informations d'accès au service, dans un annuaire de services Web.

- L' annuaire de service rend disponible l'interface du service ainsi que ses informations d'accès, pour n'importe quel demandeur potentiel de service(peut être public ou privé).

- Le consommateur de service accède à l'annuaire de service pour effectuer une recherche afin de trouver les services désirés. Ensuite, il se lie au fournisseur pour invoquer le service.

Architecture SOA



Source: http://e-technologymanagement.com/tm/articles/soa.pdf

Apports de SOA

- Améliorer la rapidité ainsi que la productivité des développements.
- Une réutilisabilité possible des services.
- De meilleures possibilités d'évolution.
- Une maintenance facilitée.
- Couplage faible entre les services.
- Architecture basée sur des standards ouverts.
- L’indépendance par rapport aux aspects technologiques.
- Une modularité permettant de remplacer facilement un service par un autre.

Références

[1] rpc-arch.png (1684×1042)
[2] corba-architecture-slide1.png (960×540)
[3] http://modelseverywhere.wordpress.com/2010/11/04/model-driven-integration/
[4] https://www.coursera.org/learn/service-oriented-architecture/
[5] http://design-patterns.fr/introduction-a-la-_programmation-orientee-objet
[6] http://blog.xebia.fr/2009/04/29/soa-du-composant-_au-service-lautonomie
[7] https://www.ibisc.univ-evry.fr/~tmelliti/cours/CPAR/cours6.pdf]
[8] http://adslbox.free.fr/rapports/rapport-gl-service-oriented-architecture.pdf
[9] http://deptinfo.unice.fr/~baude/WS/cours_SOA_AO+FB.pdf
[10] rpc1.png (720×540)