Architecture Microservices : brièvement

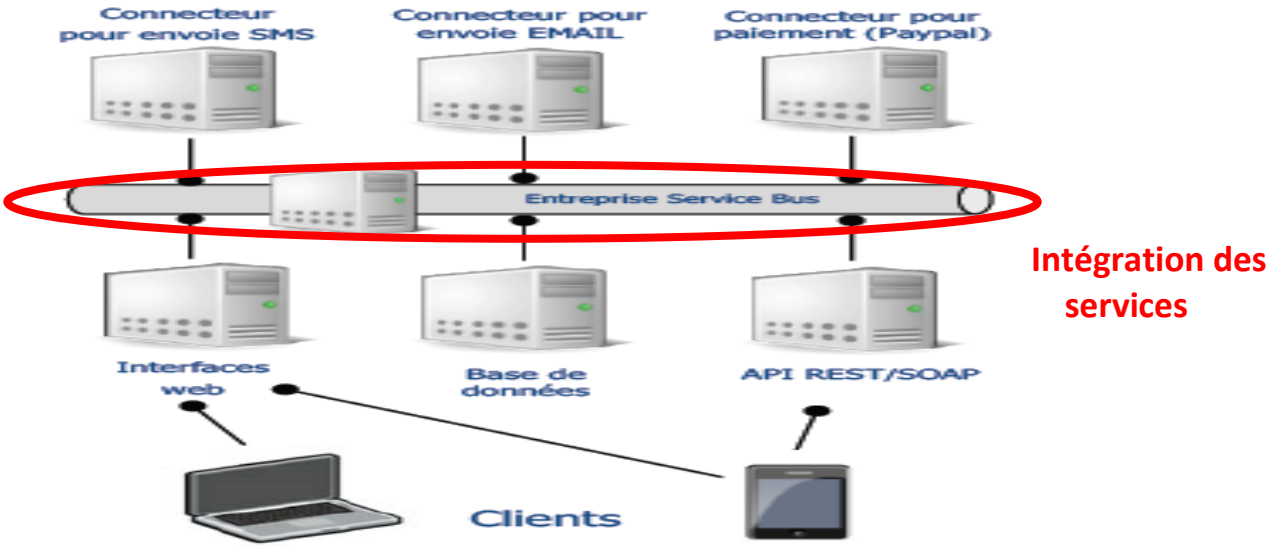
Chronologie : Architecture Monolithique => SOA=>MSA

1. Architecture Monolithique

- Un gros code contenant toutes les fonctionnalités et les différentes couches logicielles
- Une seule grosse compilation et un seul livrable (un gros fichier WAR)
- Une seule pile logicielle (Linux, JVM, Tomcat et bibliothèques tierces)

2. SOA (Service Orineted Architecture)

SOA est un style architectural qui permet de construire des solutions d'entreprises basées sur les services. Le service est une action exécutée par un **fournisseur** à l'attention d'un **client**, cependant l'interaction entre client et fournisseur est faite par le biais d'un médiateur (qui peut être un **Bus**) responsable de la mise en relation des composants.



2.1 ESB (Enterprise Service Bus)

ESB est l'une des techniques utilisées pour l'intégration des applications et des services qui n'ont pas été conçues pour fonctionner ensemble. L'ESB est un ensemble d'outils qui permettent des échanges de données sécurisées entre plusieurs services et applications différentes.

L'ESB possède quatre composants:

- Le message-oriented middleware (MOM) qui permet l'échange de messages de manière asynchrone. L'ESB utilise généralement une file d'attente pour sauvegarder les messages reçus avant d'être consommé par le destinataire.
- Les services web qui permettent d'exposer les fonctionnalités des applications existantes et d'assurer la communication avec le bus.
- Les transformations des messages.
- Le routage intelligent des messages.

Par exemple, l'ESB peut s'appuyer sur les standards suivants :

3. JMS (Java Message Services) pour implémenter MOM.
4. **SOAP, WSDL, UDDI** pour les services Web.
5. JBI (Java Business Integration) pour les conteneurs de services.
6. XML, XSLT, XPath pour la transformation et le routage.
7. JCA (Java Connector Architecture) pour la connexion aux applications.
8. BPEL pour l'orchestration des processus métier.

2.2 Principe de SOA

L'aspect le plus important de l'architecture SOA est qu'elle permet de séparer l'implémentation du service de son interface.

L'architecture orientée services est une nouvelle vision pour le système informatique. Ce dernier n'est plus décrit comme un ensemble d'applications mais comme un ensemble de services. Donc, plutôt que de privilégier une architecture applicative basée sur des contraintes techniques, l'architecture orientée service (SOA) propose de découper les fonctionnalités d'une application en services métier, réutilisables dans d'autres applications. En se concentrant sur les services, les applications sont agrégées pour fournir des processus opérationnels plus riches et plus significatifs.

Les services d'une architecture SOA répondant, notamment, aux critères suivants:

- **Faiblement couplés** : les applications traditionnelles incluent dans leur code les données métiers de l'entreprise. Elles sont complètement liées aux systèmes pour lesquels elles ont été conçues. Cette contrainte implique la difficulté de toute demande de modification, qu'elle concerne l'accès aux données, les règles de gestion ou celles de présentation. Un faible couplage permet une scission des aspects métiers du code qui permettra une simple reconfiguration des processus quand les fonctions métiers évoluent.

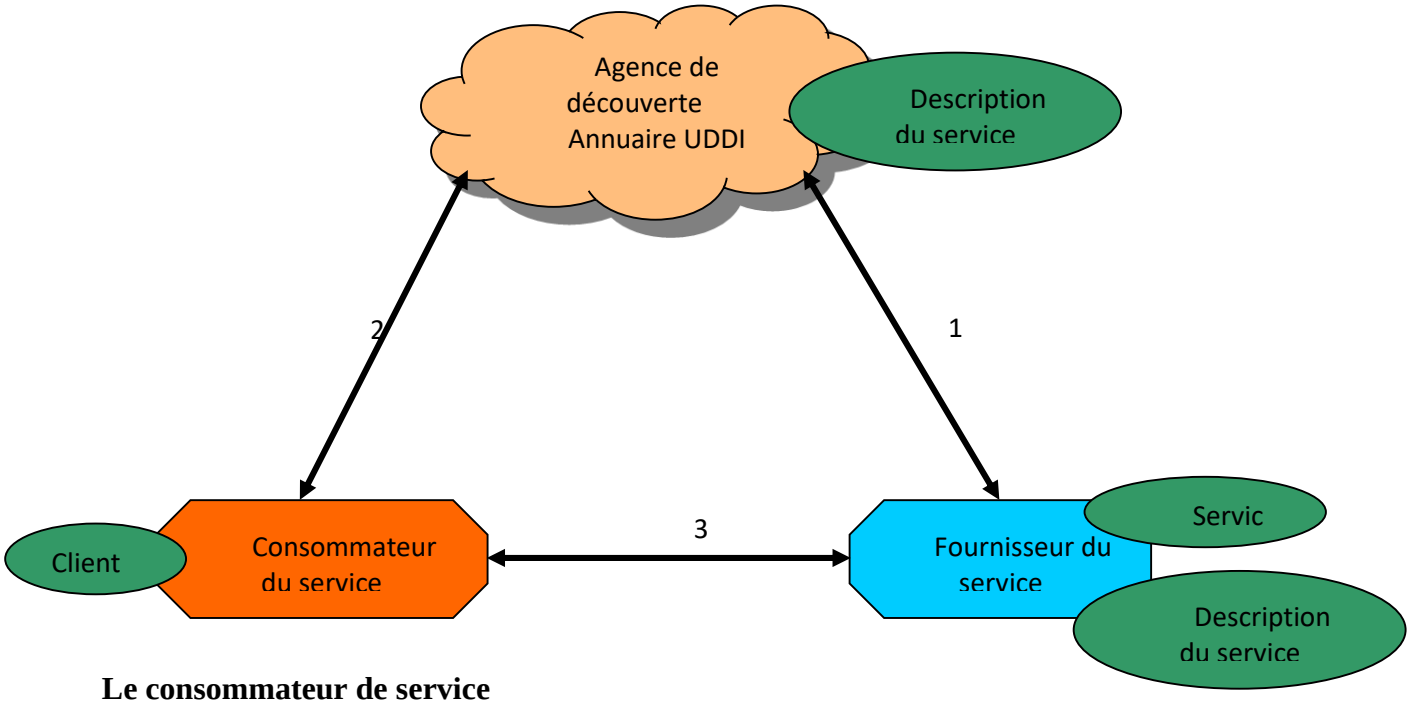
- **Distribués** : les services qui composent les applications peuvent être physiquement répartis sur des différents systèmes dans l'entreprise, mais aussi au-delà.
- **Invocables et publiables :** les services doivent être invocables et publiables quels que soit les systèmes utilisés.

2.3. Les composants de la SOA

Une architecture de services (SOA) est constituée de trois (ou 4) composants primaires. Le premier est le prestataire de services (le service réel). Vient ensuite le demandeur du service, autrement dit le composant qui accède au service. Enfin, l'agence de services fournit des services de découverte et d'enregistrement.

Le paradigme "découvrir, interagir et exécuter" comme montré dans la figure ci-dessous. Ce paradigme permet au consommateur du service (client) d'interroger un **annuaire** pour le service qui répond à ses critères. Si l'annuaire possède un tel service, alors il renvoie au client le **contrat** du service voulu ainsi que son adresse.

Le schéma suivant décrit l'architecture de base des services Web, et explique le protocole de découverte, de recherche et d'invocation des services en utilisant les trois standards de base **SOAP, WSDL et UDDI**.



Le consommateur de service

Le consommateur de service est une application qui requière un service. C'est l'entité qui initie la localisation du service dans l'annuaire, interagit avec le service à travers un protocole et exécute la fonction exposée par le service.

Le fournisseur de service

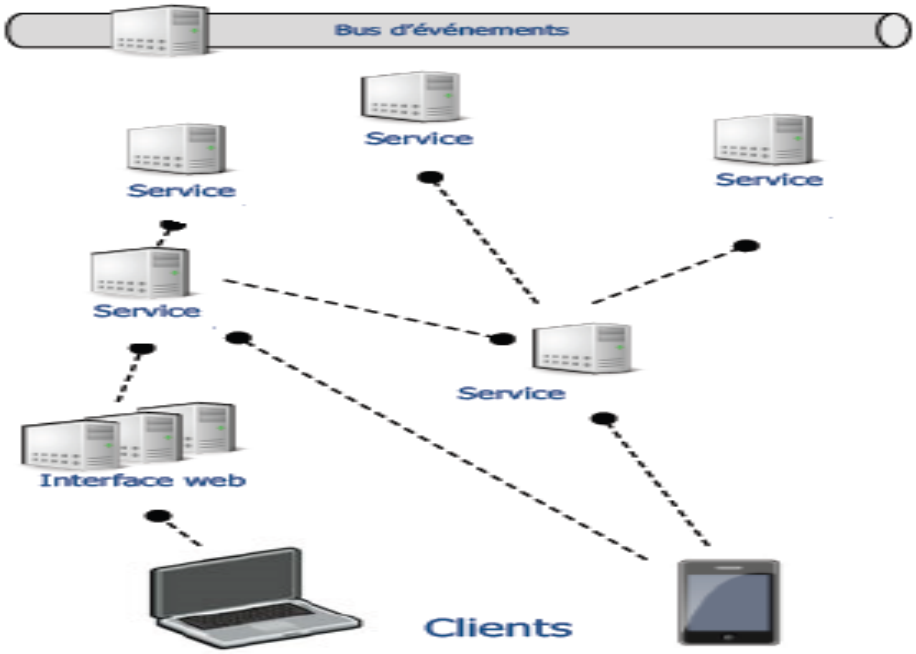
Le fournisseur de service est une entité adressable via un réseau, il accepte et exécute les requêtes venant d'un client.

Le fournisseur de service publie le contrat de service dans l'annuaire pour qu'il puisse être accédé par les clients.

L'annuaire de service

L'annuaire de service est un annuaire qui contient les services disponibles. C'est une entité qui accepte et sauvegarde les contrats du fournisseur de service et présente ces contrats aux éventuels clients.

3. L'architecture Micro-services (MSA)



- Premières discussions autour du terme microservice en 2011 lors d'un workshop sur les architectures logicielles
- Les architectures microservices sont une évolution, une spécialisation des architectures orientées services (SOA)
- Vu comme une bonne pratique du SOA (réutilisabilité)
- Une architecture microservice est donc un ensemble de microservices autonomes
- Pas de bus d'intégration (ESB) ou une orchestration BPEL par exemple (un genre de couplage même s'il contient très peu de la logique)=> Utilisation d'un Bus d'événements pour MSA => **Mais il faut équilibrer la charge** (Utilisation d'un Load Balancer)

Caractéristiques

- **Fonctionnalité unique**

  - Un microservice doit réaliser une seule fonctionnalité de l'application globale (une tache bien précise)
  - Un microservice peut contenir toutes les couches logicielles (IHM [ev front end], middleware et base de données)
  - Un microservice possède un contexte d'exécution séparé des autres (exemple : machine virtuelle ou conteneur)

- **Flexibilité technologie :** Utiliser les bons langages et frameworks selon la fonctionnalité à réaliser.

- **Equipe de développement réduite**

  - Chaque microservice à sa propre équipe de développement
  - L'équipe de développement orientée fonctionnalité (spécialisée)=> développement agile et rapide.
    - ✓ Développeurs backend
    - ✓ Développeurs web
    - ✓ Administrateurs bases de données
    - ✓ ……

- Déploiement ciblé

  - Evolution d'une certaine partie sans tout redéployer
  - Un seul livrable à partir d'un seul code source
  - Moins de coordination entre équipe quand il y a un seul déploiement
    - ✓ Déploiement plus fréquent
    - ✓ Moins de risque
    - ✓ Plus rapide
    - ⇨ **Faciliter les mises à jour :** l'entreprise est capable de faire évoluer son application de façon très fréquente et de répondre rapidement aux nouvelles fonctionnalités que propose la concurrence=>Pas nécessaire de re-tester et de déployer l'application complète.
    - ⇨ **Passer d'une technologie à l'autre sur des portions individuelles**

- Montée en charge « scalabilité » : **capable de s'étendre sur plus de ressources**
  - Forte sollicitation sur un microservice : page web d'Amazon en période de « Black Friday » par exemple
  - Possibilité de **multiplie**r le nombre d'instances d'un microservice fréquemment demandé (Comme chaque fonctionnalité est isolée).
  - Il faut donc envisager, dès le départ, une autre approche dans la conception de l'application afin que celle-ci soit "**Cloud-native**", c'est-à-dire conçue pour fonctionner et profiter pleinement des avantages qu'offre le cloud.

  - Tests facilités : Comme chaque fonctionnalité est isolée dans un microservice il est plus facile de les tester.
    - ✓ Tester l'API exposée du microservice

- ✓ Bout-en-bout : intégrer les autres microservices

†

SOA vs MSA

SOA	MSA
✓ Standards définis par OASIS SOAP, BPEL ✓ Couplage faible Mais: **dans une SOA, il est courant de trouver une base de données commune à plusieurs services (**modernisation des anciens systèmes**)**. ✓ Bus d'intégration ✓ Plusieurs fonctionnalités par service ✓ Déploiement monolithique ✓ La **SOA** accepte que ses composants (services et autres) communiquent avec des **protocoles différents**, (l'ESB s'occupant ensuite d'adapter et de transformer).	✓ Standards du web ✓ Couplage très faible : **chaque Microservice doit être très indépendant à tous les niveaux (**Chaque Microservice possède sa propre base de données**)**. ✓ Bus d'événement ✓ Une seule fonctionnalité par microservice ✓ Automatiquement déployable ✓ La **MSA** tend à obliger à utiliser **un seul protocole** et de s'y tenir.