**Revue de littérature**

Pour parvenir à offrir à PH une solution de réservation en ligne en temps réel et répondant à ses besoins, il est capital pour nous de bien comprendre les concepts théoriques liés au projet. Ainsi nous aborderons les notions comme système d’information, intégration de solution, API, paiement en ligne, Bases de données relationnelles, ...

* 1. **Généralité sur l’intégration de système d’information**
     1. **Système d’information**
        1. **Définition**

Un **système d’information** (SI) est (d’après *Wikipédia*) ensemble organisé de ressources matérielles, humaines et logicielles permettant de la collecte, le stockage, le traitement, la diffusion ou la distribution de l’information dans un environnement donné.

En outre, il s’agit d’un ensemble matériels ou immatériels (hommes, machines, méthodes, règles) en interaction transformant en processus des éléments (les entrées) en d’autres éléments (sorties).

* + - 1. **Finalité et rôle d’un Système d’Information (pour l’entreprise)**

Le système d’information tient une importance capitale dans le processus stratégique de toute entreprise. C’est en effet lui qui permet de réaliser toutes les actions liées à la gestion de données : collecte, stockage, traitement et partage… autant d’étapes qu’il rend possible à travers un ensemble de ressources à la fois matérielles, humaines et technologiques.

Détenant à la fois un rôle opérationnel et stratégique, le système d’information, au-delà de sa fonction « opérationnelle », dispose d’une véritable dimension sociale en permettant aux différents acteurs d’une entreprise de disposer de l’information dont ils disposent en [temps réel](https://www.lesjeudis.com/industries/temps-r%C3%A9el), au bon format et diffusable auprès de tous. Lorsqu’il est correctement exploité, le système d’information représente à la fois un gain de temps et d’argent, en rendant automatiques des processus jusque-là effectués manuellement mais il optimise également la performance commerciale en améliorant la relation client.

Le système d’information d’une entreprise est souvent considéré comme sa « *colonne vertébrale* », à la fois performante et évolutive. C’est pourquoi tout ce qui se touche de près ou de loin à sa conception doit faire l’objet d’une réflexion stratégique en amont du lancement du projet.

* + - 1. **Composants d’un SI**

Le domaine des systèmes d'information et de communication a certes une forte composante technologique et [informatique](https://fr.wikipedia.org/wiki/Informatique). Mais c'est seulement un aspect de ce domaine qui est en fait beaucoup plus vaste. Il s'agit de concevoir comment circule et est stockée l'information de façon efficace et cohérente pour toutes les activités d'une entreprise, d'un réseau d'entreprises, d'une administration publique, des relations entre entreprises, des citoyens, des gouvernements, etc.

Ainsi peuvent être inclus comme composants dans un système d’information en vue d’offrir des caractéristiques techniques ou fonctionnalités spécifiques :

* Applications métier ;
* [Bases de données](https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es) de l'entreprise,
* Contrôle d'accès,
* Dispositifs de [sécurité](https://fr.wikipedia.org/wiki/S%C3%A9curit%C3%A9_des_syst%C3%A8mes_d%27information),
* [Infrastructure réseau](https://fr.wikipedia.org/wiki/R%C3%A9seau_(informatique)),
* [Postes de travail informatique](https://fr.wikipedia.org/wiki/Poste_de_travail),
* Accès aux réseaux [Internet](https://fr.wikipedia.org/wiki/Internet), [intranet](https://fr.wikipedia.org/wiki/Intranet) ou [extranet](https://fr.wikipedia.org/wiki/Extranet),
* [Serveurs d'application](https://fr.wikipedia.org/wiki/Serveur_d%27applications),
* Serveurs de [données](https://fr.wikipedia.org/wiki/Donn%C3%A9e) et systèmes de [stockage](https://fr.wikipedia.org/wiki/Sauvegarde),
* Système de paiement électronique,
* Système de sécurité (protection et [chiffrement](https://fr.wikipedia.org/wiki/Chiffrement)),
* Outils de [groupware](https://fr.wikipedia.org/wiki/Groupware), agendas, espace de partage de documents, échange d'informations (forums électroniques), gestion de contacts, conférence électronique (chat, vidéoconférence).
  1. **Intégration de système d’information**

L’**intégration de système** consiste à réunir au sein d’un même systèmes d’information, des parties développées de façon séparées

* + 1. **Présentation**

L'***intégration informatique***, ou ***intégration des systèmes***, désigne le fait de connecter les données, les applications, les API et les appareils au sein d’un service informatique, dans le but d'augmenter l'efficacité, la productivité et l'agilité d’une entreprise. Elle permet à tous les éléments d'un environnement informatique de fonctionner ensemble. Ainsi, elle représente un aspect clé de la transformation d'une entreprise, c'est-à-dire de son adaptation face à l'évolution du marché.

L'intégration ne se limite pas à connecter des éléments, elle ajoute également de la valeur. En effet, en connectant les différentes fonctions de plusieurs systèmes, elle peut offrir de nouvelles fonctionnalités.

***Intégration système différent d’intégration continue***

***L’intégration continue*** *est* pratique employée par les développeurs, consistant à fusionner des versions de codes qui fonctionnent sur un référentiel central partagé plusieurs fois par jour. L'objectif de l'intégration continue est d'automatiser la création de versions et la vérification afin de détecter rapidement les éventuelles erreurs et d'accélérer le développement.

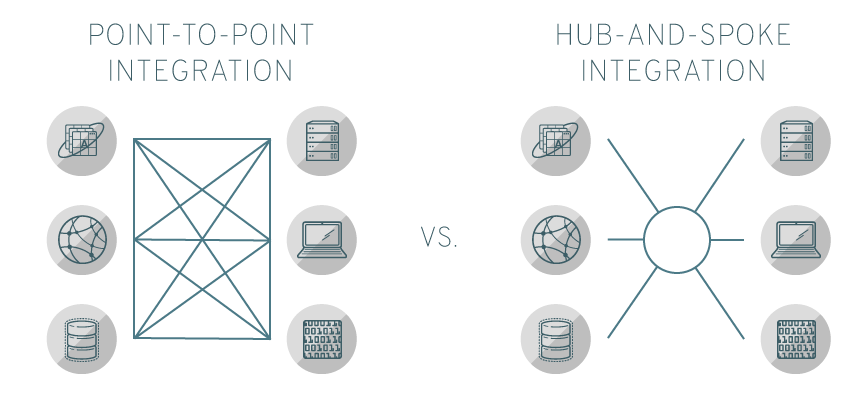
* + 1. **Histoire de l’intégration**

Au fil du temps, les systèmes informatiques grossissaient et se développaient, ils s’éloignaient les uns des autres. Les solutions provenant de fournisseurs différents ne pouvaient plus communiquer entre elles. Au final, les éléments d’une pile informatique n'étaient plus connectés que par le simple fait qu'ils appartenaient à une organisation. Il fallait donc trouver une solution pour organiser l’environnement et éviter la répétition des tâches, surtout lorsqu'il s'agit de mettre en œuvre et de gérer une logique métier.

* + 1. **Les solutions d’intégration de systèmes** 
       1. **L’intégration d’application d’entreprise**

Il existe une solution pour rassembler et organiser tous ces éléments disparates **: l'intégration d'applications d'entreprise** (EAI), c'est-à-dire un ensemble de technologies, des outils et un Framework qui permettent une intégration en temps réel et basée sur les messages entre les différentes applications. L'envoi de ces messages est déclenché par des modifications ou des paramètres au sein de chaque application.

Il existe deux modèles d'EAI : l'intégration point à point (« ***point-to-point*** ») et l'intégration en étoile (« ***hub-and-spoke*** »).

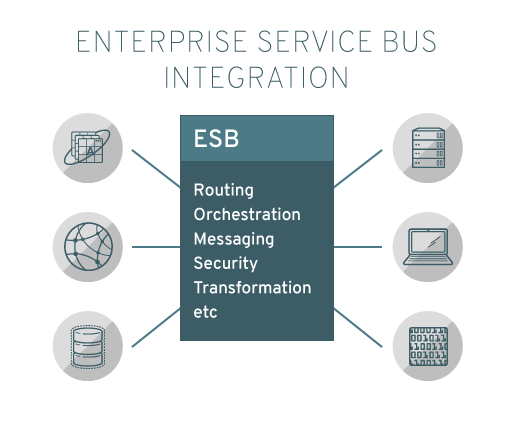


*Point-to-point vs Hub-and-spoke integration*

* **Le modèle point à point :** ici, chaque application (ou ressource) est personnalisée pour pouvoir communiquer (fonctionner) avec les autres applications et éléments de l’environnement informatique. Il s'agit d'une tâche fastidieuse et le modèle est par conséquent fortement sujet aux erreurs. Pour ne rien simplifier, la maintenance du modèle se complexifie à chaque mise à jour de l'infrastructure et des applications.
* **Le modèle étoile :** Il résout le problème du modèle point à point avec un point de connexion central (le cœur) qui relie toutes les applications et les services. Les rayons qui relient le cœur aux applications et aux services peuvent faire l'objet d'une maintenance individuelle. Ainsi, il est possible de concevoir des applications plus spécialisées et réserver les tâches d'intégration au cœur et aux rayons. Le principal désavantage de cette approche réside dans la centralisation du cœur, car il devient le point unique de défaillance du système et des communications au sein de votre infrastructure. Dans un modèle en étoile, toutes les intégrations dépendent, par définition, du bon fonctionnement du cœur.
  + - 1. **ESB**

**ESB** pour *Enterprise Service Bus* est un outil d'abstraction orienté messages qui fournit des modules de service entre les applications.

L'ESB tient également le rôle de point central où tous les modules de service sont partagés, redirigés et organisés afin de connecter les applications et données entre elles. Cette solution remplace avantageusement l'intégration en étoile, mais ce n'est pas non plus la panacée, surtout lorsque l’entreprise croît et acquiert de nouvelles ressources, car elle requiert alors une plus grande rapidité au niveau des propriétés et des ressources logicielles.



*ESB intégration*

De manière structurelle, un ESB ressemble fort au modèle d'intégration en étoile. Mais l'ESB propose tout de même quelques fonctionnalités très spécifiques.

* Un ESB est un service qui utilise des normes ouvertes. Ainsi, pas besoin d'écrire des interfaces uniques pour chaque application.
* Les services d'intégration peuvent être déployés sans modifier fondamentalement les applications.
* Les ESB s'appuient sur des interfaces et des protocoles ouverts et standard, afin de simplifier les nouveaux déploiements.

Toutefois, un déploiement d'ESB implique souvent une architecture centralisée, et ce, pour les mêmes raisons évidentes déjà avancées avec le modèle en étoile : il faut un emplacement unique pour héberger et contrôler tous les services d'intégration. Malheureusement, les déploiements et architectures d'ESB s'accompagnent d'une gouvernance centrale rigide qui ne permet pas de fournir les solutions plus rapides et capables de s'adapter qui sont à la base des initiatives de [transformation numérique](https://www.redhat.com/fr/topics/digital-transformation/what-is-digital-transformation). De plus, les ESB finissent souvent par se changer en applications monolithiques.

* + - 1. **Intégration agile**

Aujourd'hui, les architectures d'applications **cloud-native**, faiblement couplées et développées selon des méthodes [agiles](https://www.redhat.com/fr/devops/what-is-agile-methodology) et des principes DevOps, exigent une approche évolutive et tout aussi agile de l'intégration. L'intégration agile (Selon *Red Hat*) est une stratégie de connexion des ressources qui allie des technologies d'intégration, des techniques de distribution agile et des plateformes natives pour le cloud dans le but d'accélérer la distribution des logiciels tout en renforçant la sécurité. En pratique, l'approche consiste à déployer des technologies d'intégration (des API, par exemple) dans des conteneurs Linux, et à répartir la gestion de l'intégration entre des équipes pluridisciplinaires. Une architecture d'intégration agile peut être divisée en trois capacités clés : **l'intégration distribuée**, les [**conteneurs**](https://www.redhat.com/fr/topics/containers)et les **API**.

|  |  |  |
| --- | --- | --- |
| **Intégration distribuée** | **Les conteneurs** | **Les API (Application Programming Interface)** |
| * Encombrement réduit * Basée sur des modèles * Orientée événements * Communautaire   Avantage : **FLEXIBILITÉ** | * Cloud-native * Simples et déployables individuellement * Évolutifs et hautement disponibles   Avantage : **ÉVOLUTIVITÉ** | Points de terminaison bien définis, réutilisables et bien gérésInfluence et utilisation de l'écosystème Avantage : **RÉUTILISATION** |

* + - 1. **Architecture orienté évènements**

**L'architecture orientée événements** (de l'anglais *event driven architecture*, ou **EDA**) est une forme d'[architecture de médiation](https://fr.wikipedia.org/wiki/Architecture_de_m%C3%A9diation) qui est un modèle d'interaction applicative mettant en œuvre des services (composants [logiciels](https://fr.wikipedia.org/wiki/Logiciels)) répondant à des sollicitations externes :

* avec une forte cohérence interne (par l'utilisation d'un format d'échange pivot, le plus souvent [XML](https://fr.wikipedia.org/wiki/XML)),
* Et des couplages externes lâches (par l'utilisation d'événements)

Par opposition à l'[architecture orientée services](https://fr.wikipedia.org/wiki/Architecture_orient%C3%A9e_services) (SOA) où un « fournisseur » rend un service à la demande d'un consommateur; en architecture EDA, un [composant](https://fr.wikipedia.org/wiki/Composant_logiciel) prévient par émission d'un événement qu'il a réalisé une opération donnée. C'est aux Clients potentiels de traiter cet événement.

Les architectures EDA ont été popularisées avec l'apparition de standards pour les places de marchés et les systèmes de vente aux enchères. Elles mettent en application une partie des principes d'[urbanisation](https://fr.wikipedia.org/wiki/Urbanisation_(informatique)). Une architecture orientée événements repose principalement sur un bus disposant de fonctionnalités d'abonnement et de publication (*publish and Subscribe*)

***1.2.3.4.1 Les concepts de l’architecture orienté évènements***

Le couplage entre composants est un couplage lâche et les communications sont en général [asynchrones](https://fr.wikipedia.org/wiki/Asynchronisme).

*Le service peut :*

* Être codé dans n'importe quel langage,
* S’exécuter sur n'importe quelle plateforme (matérielle et logicielle).

*Le service doit :*

* S’abonner aux événements qu'il souhaite traiter
* Traiter les événements auquel il est abonné sans préjuger d'un quelconque ordre et émettre un événement compte rendu de l'action qu'il vient de réaliser
* Fournir les événements qu'il est susceptible d'émettre dont les structures sont publiées,
* Être autonome (disposer de toutes les informations nécessaires à son exécution : pas de notion d'état)
* Respecter un ensemble de contrats (règles de fonctionnement).

***1.2.3.4.2 Un évènement***

Un **événement** désigne tout phénomène ou changement d'état significatif au niveau du matériel ou d'un logiciel système. Il ne faut pas confondre un événement et une notification d'événement, c'est-à-dire une notification ou un message envoyé par le système pour signaler à une autre partie du système qu'un événement s'est produit.

Les événements peuvent être causés par des actions internes ou externes. Ils peuvent être provoqués par des utilisateurs (clics de souris ou frappe sur le clavier, par exemple), provenir d'une source externe (un capteur) ou être générés par le système (lors du chargement d'un programme, par exemple).

Les consommateurs d'un événement doivent s'abonner (*subscribe*) à un intermédiaire de gestion d'événement (le bus) et le producteur de l'événement doit le publier auprès de ce gestionnaire (le bus). Quand le gestionnaire d'événement reçoit un événement d'un producteur, il le diffuse (*forward*) aux consommateurs concernés. Si le consommateur est injoignable, le gestionnaire peut conserver le message et le diffuser ultérieurement. Ce moyen de transmission d'événements repose sur un bus de message *store and forward*.

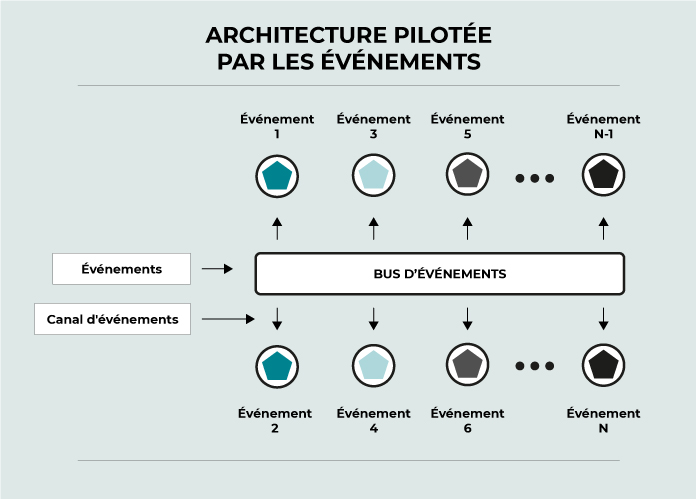
***1.2.3.4.3 Fonctionnement***

Ce type d'architecture implique des **producteurs** et des **consommateurs** d'événements. Un producteur d'événements détecte ou reconnaît un événement et le représente sous forme de message. Il ignore quels seront les consommateurs et les conséquences de chaque événement.

Lorsqu'un événement a été détecté, il est transmis du producteur d'événements au consommateur via des canaux d'événement, où une plateforme de traitement les prend en charge de façon asynchrone. Les consommateurs d'événements doivent être informés lorsqu'un événement se produit. Ils peuvent traiter l'événement ou être seulement affectés par ce dernier.

La plateforme de traitement des événements exécute la réponse adaptée à chaque événement et envoie l'activité en aval aux consommateurs concernés. Cette activité permet de visualiser le résultat d'un événement.

Il existe d'autres gestionnaires d'événements de type middleware qui peuvent servir de plateforme de traitement des événements.



***1.2.3.4.4 Modèles d'architectures orientées événements***

Une architecture orientée événements peut être basée sur un modèle de publication/abonnement ou sur un modèle de flux d'événements.

#### **Modèle de publication/abonnement**

Ce modèle est une infrastructure de messagerie basée sur des abonnements à des flux d'événements. Lorsqu'il est utilisé, chaque fois qu'un événement se produit ou est publié, il est envoyé aux abonnés qui doivent en être informés.

#### **Modèle de flux d'événements**

Avec un modèle de flux d'événements, les événements sont enregistrés dans un journal. Au lieu d'être abonnés à un flux d'événements, les consommateurs d'événements peuvent accéder à n'importe quelle partie du flux et le rejoindre à tout moment.

Il existe différents types de flux d'événements :

|  |  |  |
| --- | --- | --- |
| **Le traitement des flux d’évènements** | **Le traitement des évènements simples** | **Le traitement des évènements complexes** |
| S’appuie sur une plateforme de diffusion de données en continu, telle qu'Apache Kafka, pour ingérer des événements et des processus, voire transformer le flux d'événements. Le traitement des flux d'événements peut permettre d'y détecter des tendances importantes. | correspond aux cas où un événement déclenche immédiatement une action au niveau du consommateur d'événements. | oblige un consommateur d'événements à traiter une série d'événements afin de détecter des schémas récurrents. |

**1.2.3.4.5 Avantage de l’architecture orienté évènements**

La programmation orientée événements présente plusieurs avantages, notamment :

* **Des services faiblement connectés**: les créateurs et les consommateurs d’événements ne sont pas connectés. Le créateur ne sait pas quels consommateurs attendent les événements qu’il produit. Il envoie simplement l’événement à un routeur, où les consommateurs pourront trouver les types d’événements auxquels ils sont abonnés.
* **Évolutivité**: comme les créateurs et les consommateurs ne se connaissent pas, il n’y a pas d’intégration directe. Il est beaucoup plus facile d’ajouter de nouveaux services dans le système et d’éviter les pannes au cours de cette opération.
* **Développement agile** : comme il est facile d’ajouter de nouveaux services aux systèmes EDA, le développement s’intègre bien dans un [environnement agile](https://www.lucidchart.com/blog/fr/release-planning-agile). Le routeur d’événements est le médiateur entre les services qui filtre et envoie les événements aux consommateurs.
* **Réduction des coûts** : les systèmes EDA sont basés sur le principe du « push », à la demande. Le système n’interroge pas continuellement les serveurs. La consultation et le traitement à la demande requièrent moins de ressources réseau, ce qui réduit les coûts globaux.
* **Réponse et analyses en temps réel** : les applications réagissent aux changements en temps réel à l’aide des données actuelles et historiques pour prendre des décisions. Ce type de réponse aide le système à prévoir et à éliminer les menaces de sécurité à la volée, en automatisant et en optimisant les chaînes d’approvisionnement.

**1.2.3.4.6 Pourquoi utiliser l’architecture orienté évènements**

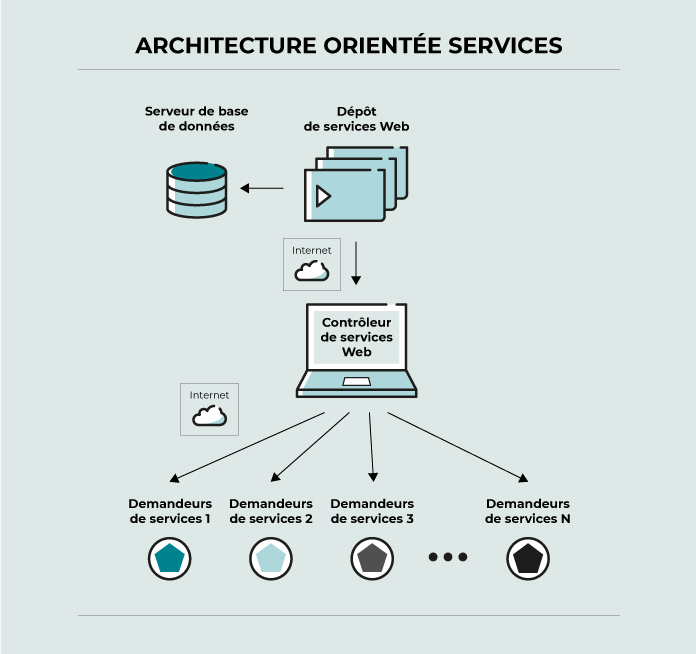
Lorsqu’on souhaite que divers systèmes ou sous-système soient interopérables, l’utilisation d’un EDA est utile pour répondre aux besoins suivants :

* **Réplication des données entre comptes et entre régions :** Les routeurs d’événements peuvent transférer des données vers différents systèmes situés dans différentes régions et différents comptes. Cette approche permet de garantir la synchronisation des systèmes tout en laissant vos équipes développer et déployer de nouvelles applications et de nouveaux services indépendamment des autres équipes.
* **Répartition et traitement parallèle** : utilisez l’architecture EDA pour « répartir » un événement unique entre plusieurs systèmes qui doivent y répondre, ce qui évite à vos équipes de devoir développer un code personnalisé pour transmettre l’événement à chaque consommateur. Chaque système peut traiter l’événement en même temps à des fins différentes.
* **Surveillance et avertissements de l’état des ressources** : les architectures EDA peuvent surveiller les ressources et vous envoyer des alertes en cas de changement d’état, d’anomalies et de mise à jour des composants.
* **Intégration de systèmes hétérogènes** : utilisez des routeurs d’événements pour partager des informations entre des systèmes utilisant différentes piles technologiques.
  + - 1. **Architecture orienté Services (SOA)** 
         1. **Définition**

L'architecture orientée services (ou SOA, *Service-Oriented Architecture*) est un modèle de conception qui rend des composants logiciels réutilisables, grâce à des interfaces de services qui utilisent un langage commun pour communiquer via un réseau.

Un **service** est une unité autonome de fonctionnalité logicielle, ou d'un ensemble de fonctionnalités, conçue pour réaliser une tâche précise comme récupérer des informations ou exécuter une opération. Il contient les intégrations de code et de données nécessaires pour exécuter une fonction métier distincte et complète. Vous pouvez y accéder à distance, et interagir avec lui ou le mettre à jour de manière indépendante.  
En d'autres termes, l'architecture SOA permet à des composants logiciels déployés et gérés séparément de communiquer et de fonctionner ensemble sous la forme d'applications logicielles communes à différents systèmes.

* + - * 1. **Comment fonctionne SOA ?**



*Architecture SOA*

Au sein d'une architecture orientée services, les services communiquent via un système de « couplage faible ». Les composants (aussi appelés « éléments ») sont interconnectés dans un système ou réseau, ce qui leur permet de transmettre des informations ou de coordonner un processus métier, en réduisant la dépendance entre eux.

L’architecture Orienté services est structuré en quatre parties :

* **Le dépôt de services web (Web service repository) :** Il s'agit d'une bibliothèque de services web conçue pour répondre à des demandes d'informations externes. L'information fournie est généralement un petit élément, comme un numéro, un mot, quelques variables, etc. Par exemple, un numéro de vol, un numéro de suivi de colis, le statut d'une commande (une lettre), etc. Cette bibliothèque est généralement documentée de manière très détaillée, car des applications externes font appel aux fonctions qu'elle contient.
* **Le contrôleur de services web (Web service controller) :** Ce module communique les informations contenues dans le dépôt de services web aux demandeurs de services. Lorsqu'un demandeur de service externe appelle une certaine fonction du dépôt de services web, le contrôleur de services web interprète la demande et recherche la fonction dans le dépôt de services web. Il exécute ensuite cette fonction et renvoie une valeur au demandeur.
* **Le serveur de base de données (Database Server) :** Ce serveur contient les tables, les index et les données gérés par l'application. Les recherches et les opérations d'insertion/suppression/mise à jour sont exécutées ici.
* **Les demandeurs de services (Service Requester) :** Il s'agit d'applications externes qui demandent des services au dépôt de services web par l'intermédiaire d'Internet, comme une organisation demandant des informations sur les vols à une compagnie aérienne, ou une autre entreprise demandant à un transporteur la localisation d'un colis à un moment donné.
  + - * 1. **Avantages par rapport à une approche monolithique**
* **Mise sur le marché accélérée et plus grande flexibilité :**le caractère réutilisable des services facilite et accélère le regroupement des applications. Les développeurs ne doivent plus repartir de zéro à chaque fois, comme c'est le cas pour les applications monolithiques.
* **Utilisation de l'infrastructure existante sur les nouveaux marchés :**grâce à l'architecture SOA, les développeurs peuvent plus facilement étendre et mettre à l'échelle les fonctionnalités d'une plateforme ou d'un environnement.
* **Coûts réduits grâce à une meilleure agilité et à un développement plus efficace**
* **Maintenance facilitée :**les services étant autonomes et indépendants, ils peuvent être modifiés et mis à jour autant que nécessaire, sans affecter les autres services.
* **Évolutivité :**comme l'architecture SOA s'adapte à plusieurs services, plateformes et langages de programmation, l'évolutivité est considérablement accrue. En outre, le protocole de communication standardisé limite les interactions entre les clients et les services pour les entreprises, en diminuant le degré d'interaction, les applications peuvent être mises à l'échelle plus facilement et avec moins de pression.
* **Fiabilité améliorée :**il est plus facile de déboguer des petits services plutôt qu'un long code, ce qui permet de créer des applications plus fiables.
* **Grande disponibilité :**les fonctions de l'architecture SOA sont à la portée de tous.
  + - * 1. **Inconvenants de SOA**
* Les services web peuvent représenter une faiblesse au niveau du site pour les pirates qui veulent engorger le système. Certaines formes d'attaques sont des « dénis de service ». Elles consistent à demander le même service web des millions de fois par seconde, jusqu'à ce que le serveur tombe en panne. Il existe aujourd'hui une technologie permettant de résoudre ce problème, mais c'est toujours un problème à prendre en compte dans les architectures de services web.
* Le propriétaire du service web aide d'autres sites, mais reçoit une petite rémunération pour ce faire.
  + - * 1. **Quand utilisé SOA**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom de l'exemple** | **Définition** | **Exemple concret** | **Avantages** |
| Suivi de colis | Un service web de suivi des colis est une fonction qui, en fonction d’un numéro de suivi, décrit l'itinéraire et la localisation actuelle d'un colis commandé par un client. | * FedEx shipping Integration (fedex.com). * DHL Express Package Tracking (dhl.com). | Les sites de vente au détail qui utilisent ces services web ne doivent pas accéder aux systèmes internes de FedEx ou de DHL.  La fonction de suivi de colis réside sur Internet et peut être utilisée librement par tout site de vente au détail. |
| Informations sur les vols | Un service web d'information sur les vols accepte en entrée un numéro de vol. Il renvoie les données relatives au vol : heure de départ, heure d'arrivée prévue, lieu de départ, lieu d'arrivée, altitude actuelle, etc. | Flight Explorer Professional Edition (flightexplorer.com) | Les sites de voyage, les hôtels et les entreprises de logistique qui utilisent ce produit peuvent disposer d'informations immédiates sur un vol par Internet sans avoir à accéder aux systèmes internes. |
| Convertisseur de devises | Un service web de conversion de devises accepte une devise source et donne sa valeur dans une devise de destination (par exemple, « combien de dollars américains font un euro »). | Toutes les grandes banques en disposent, y compris la banque centrale de nombreux pays. | Les transactions monétaires nécessitent les taux de change officiels à un instant donné. Ce service web de conversion de devises publie toutes les conversions de devises fournies par la Banque centrale. |

* + - * 1. **Architecture SOA ou microservices ?**

Le concept de services introduit par l'architecture orientée services est aujourd'hui un élément essentiel du [cloud computing](https://www.redhat.com/fr/topics/cloud) et de la [virtualisation](https://www.redhat.com/fr/topics/virtualization) dans les solutions de type middleware et microservices.

On confond souvent l'architecture SOA et l'architecture de microservices en raison de leurs similitudes. Toutefois, la principale caractéristique qui les distingue est leur portée : l'architecture SOA est une approche à l'échelle de l'entreprise, tandis que les microservices sont une stratégie de mise en œuvre au sein des équipes de développement des applications.

Ils ne communiquent pas non plus de la même façon avec leurs composants. L'architecture SOA utilise un ESB, alors que les microservices peuvent communiquer entre eux sans état, à l'aide d'[API](https://www.redhat.com/fr/topics/api) indépendantes de tout langage, ce qui permet aux équipes de développement de choisir leurs outils. Ainsi, les microservices offrent plus de tolérance et de flexibilité.

Il arrive également que l'architecture SOA soit confondue avec le modèle [SaaS](https://www.redhat.com/fr/topics/cloud-computing/what-is-saas) (Software-as-a-Service). Le SaaS est une forme de cloud computing qui permet de fournir une application cloud aux utilisateurs, avec ses plateformes et son infrastructure sous-jacentes. Dans une architecture SOA, les services web sont parfois distribués sous forme d'applications SaaS. En général, un fournisseur de services cloud (par exemple AWS, Azure ou IBM Cloud) s'occupe de la gestion de l'environnement cloud sur lequel l'application SaaS est hébergée.

Les utilisateurs interagissent avec le logiciel par le biais d'un navigateur web sur leur ordinateur ou leurs appareils mobiles. Ils peuvent également le connecter à d'autres fonctions à l'aide d'API telles que REST ou SOAP.

* + 1. **Les enjeux liés à l’intégration des systèmes d’information**

L’intégration des systèmes en informatique figure aujourd’hui parmi les principaux enjeux des entreprises. En effet, dans un contexte où les données, les processus et les acteurs sont en perpétuelle évolution, il est parfois bien compliqué de parvenir à « tout faire marcher » ensemble.

Cette difficulté à réunir des éléments très hétérogènes et ne partageant bien souvent que très peu de standards, (en termes de langage de programmation, de format ou encore de [système d’exploitation](https://www.lesjeudis.com/technologies/syst%C3%A8me-d'exploitation) par exemple) met en évidence des problématiques de type technologique, économique et organisationnel auxquelles les entreprises doivent faire face. Le volume de données de plus en plus important ainsi que les [exigences en matière de confidentialité et de sécurité](https://blog.lesjeudis.com/rgpd-la-protection-des-donnees-evolue-en-2018-etes-vous-prets) transforment un peu plus la question de l’intégration des systèmes en un véritable défi.

L’intégration des systèmes permet, à terme, de répondre à différents besoins formulés par les entreprises :

* **Obtenir des processus automatiques** et donc réduire les erreurs pouvant être effectuées lors des processus « manuels » (actions de la part des utilisateurs pour faire converger les différentes applications entre elles)
* **Consolider les données**en étant capable de croiser différents types de données entre elles : en liant les données entre elle, l’entreprise a plus de visibilité en obtenant facilement des statistiques globales
* **Gagner du temps et de l’argent** en regroupant l’ensemble des informations sur un seul et même support (au lieu d’avoir à saisir ces informations sur chacune des applications)

C’est pourquoi, davantage que des enjeux techniques, ce sont bien des [enjeux stratégiques qui se cachent derrière tout projet d’intégration](https://www.supralog.com/technologie-integration-et-enjeux).

Pour l’entreprise, il s’agit de trouver la meilleure équation possible entre les besoins liés à son métier et les capacités logicielles qui sont mise à son service. La réussite d’un tel projet réside en effet en la capacité d’adopter une vision globale tout en faisant intervenir des leviers complémentaires.