

Architecture Logicielle

Séquence 2: Architecture Multi-tiers

INSA BADJI



Types d'architectures: Architecture en couches (aussi appelée architecture multi-tiers)

- L'architecture en couches (aussi appelée architecture multi-tiers) est une pratique d'architecture logicielle qui propose de concevoir le système comme une superposition de strates, chaque strate étant définie par une responsabilité spécifique.
- Cette pratique a commencé avec l'architecture client-serveur qui décrivait de quelle façon plusieurs applications ou plusieurs instances de la même application pouvaient partager la même base de données. (Voir Wikipedia [client-serveur](#))
- L'étape suivante fut de centraliser aussi les règles d'affaires dans le but de résoudre les problèmes inhérents aux applications distribuées. C'est ce qu'on a appelé l'architecture 3-tiers. (Voir Wikipedia [Architecture trois tiers](#))
- Le concept a évolué pour identifier et distinguer un nombre de couches de plus en plus nombreuses et de plus en plus fines. (Voir [L'architecture à 5 couches](#))

Architecture Multi-tiers

- Traditionnellement une application informatique est un programme exécutable sur une machine qui représente la logique de traitement des données manipulées par l'application.
- Ces données peuvent être saisies interactivement via l'interface ou lues depuis un disque. L'application émet un résultat sous forme de données qui sont, soit affichées, soit enregistrées sur un disque.
- Dans ce schéma, les traitements, les données d'entrées, les données de sortie sont sur une seule machine.

Données en
entrée

► Application

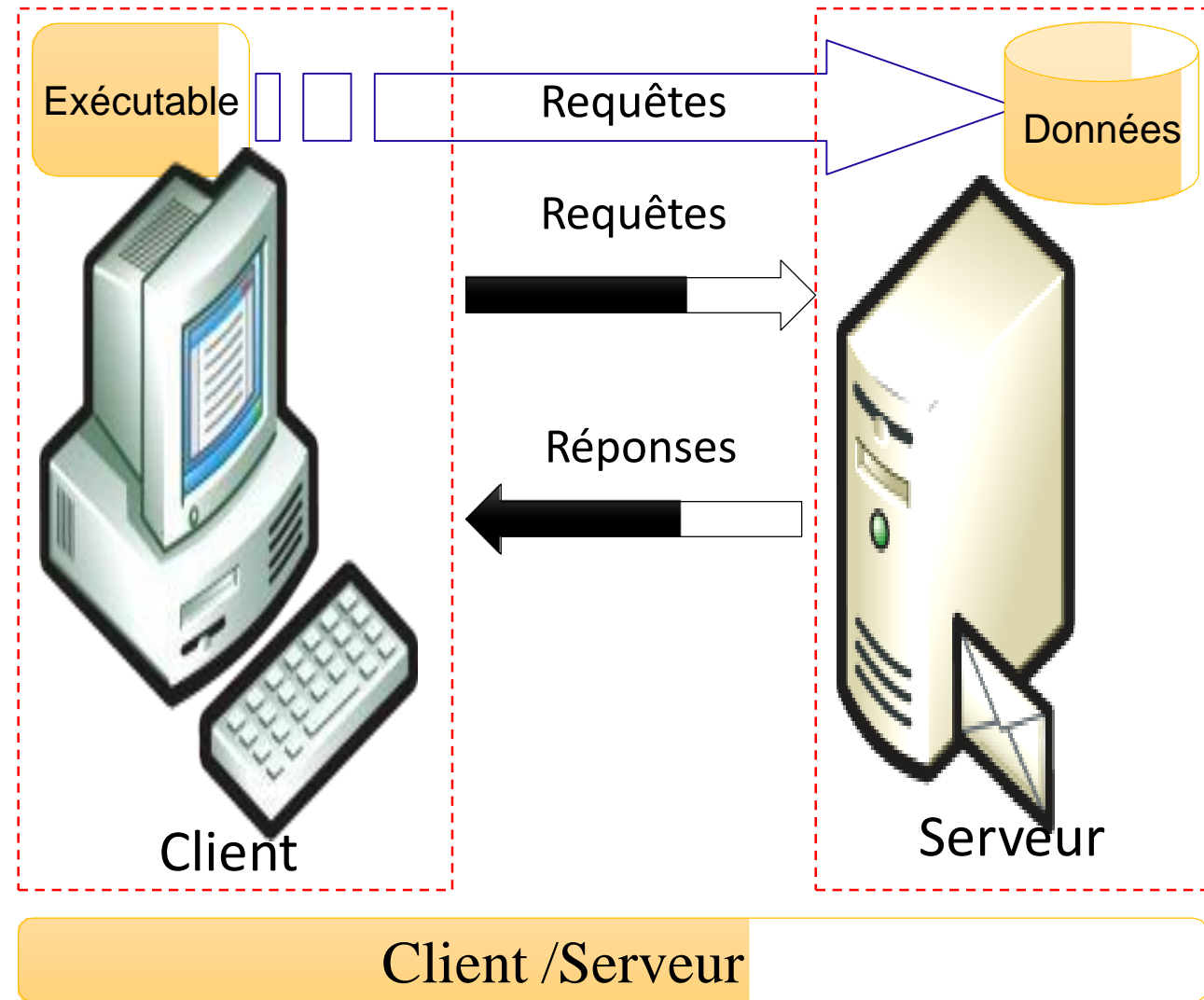
► Données en
sortie

Architecture Multi-tiers

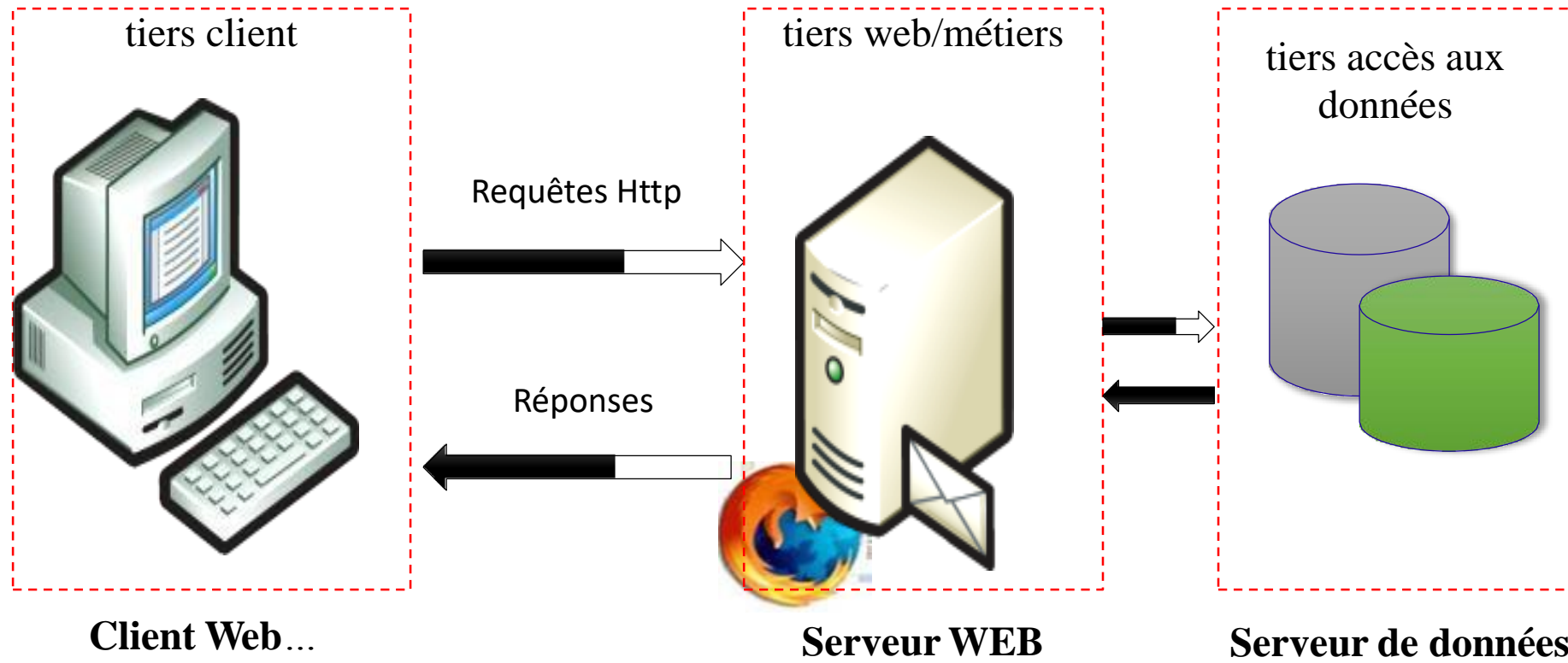
- On peut alors séparer l'application en différentes parties :
 - La couche interface homme machine
 - La couche de traitement
 - La couche de gestion des données.
- Et toute application possède ces trois parties.
- On parle de **couches**, de **niveaux** ou de **tiers** (de l'anglais tiers : étage).
- Jusqu'au années 90, ces trois couches étaient la plupart du temps sur la même machine.
- L'application utilisait les disques de la machine pour lire les données à traiter et stocker les données résultantes du traitement (sauf sur les gros systèmes).

Architecture Multi-tiers

- A partir des années 90, vu l'explosion de la volumétrie des données, on a séparé les applications des données. Ces dernières étaient alors stockées dans des bases de données sur d'autres machines.
- Les applications accédaient aux données à travers les réseaux sur des machines serveurs de données disposant d'un SGBD. C'est une **architecture 2-tiers**, ou encore **architecture client- serveur**.



Architecture 3-tiers



Architecture Client /Serveur Web : 3 tiers

Avec l'apparition des technologies Web, il est possible de séparer la **couche présentation** de la **couche applicative** (aussi appelée couche métiers)

Architecture 3-tiers : Couche Présentation

- Elle correspond à la partie de l'application visible et interactive avec les utilisateurs.
- On parle d'IHM. Elle peut être réalisée par une application graphique ou textuelle, en HTML, en WML (on parle alors de **clients légers**).
- Cette interface peut prendre de multiples facettes sans changer la finalité de l'application.

Définition. Un développeur **front-end** est un professionnel de l'informatique et/ou du web design capable de produire des sites web en utilisant ses connaissances en HTML, CSS et javascript, éventuellement couplées à des compétences dans le domaines d'un ou de plusieurs gestionnaires de contenus (CMS).



Développeur front-end (ou front office) - 1min30 - Agence web 1min30 ...
<https://www.1min30.com/dictionnaire-du-web/developpeur-front-end-ou-front-office>

Architecture 3-tiers : Couche Présentation

- Dans le cas d'un système de distributeurs de billets, l'automate peut être différent d'une banque à l'autre, mais les fonctionnalités offertes sont similaires et les services identiques (fournir des billets, donner un extrait de compte, etc.).
- Une même fonctionnalité métiers (par exemple, la commande d'un nouveau chéquier) pourra prendre différentes formes de présentation selon qu'elle se déroule sur Internet, sur un distributeur automatique de billets ou sur l'écran d'un chargé de clientèle en agence.

Architecture 3-tiers : Couche Présentation

- La **couche présentation** relaie les requêtes de l'utilisateur à destination de la **couche métiers**, et en retour lui présente les informations renvoyées par les traitements de cette couche. Il s'agit donc ici d'un assemblage de services métiers et applicatifs offerts par la couche inférieure.
- NB: 3-tiers est une application du modèle plus général qu'est le multi-tiers.

Architecture 3-tiers : Couche applicative

- Egalement **couche métiers** ou **business** ou encore **serveur d'applications**:
- Elle correspond à la partie fonctionnelle de l'application, celle qui implémente la « logique », et qui décrit les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation.
- Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche.

Architecture 3-tiers : Couche applicative

- La couche métiers offre des services applicatifs et métiers à la couche présentation. Pour fournir ces services, elle s'appuie, le cas échéant, sur les données du système, accessibles au travers des services de la couche inférieure.
- En retour, elle renvoie à la couche présentation les résultats qu'elle a calculés.

Backend. ... En informatique, un **back-end** (parfois aussi appelé un arrière-plan) est un terme désignant un étage de sortie d'un logiciel devant produire un résultat. On l'oppose au front-end (aussi appelé un frontal) qui lui est la partie visible de l'iceberg.

Backend — Wikipédia

<https://fr.wikipedia.org/wiki/Backend>

Architecture 3-tiers : Couche données

- Elle consiste en la partie gérant l'accès aux gisements de données du système.
- Ces données peuvent être **propres au système**, ou **gérées par un autre système**.
- La couche métiers n'a pas à s'adapter à ces deux cas, ils sont transparents pour elle, et elle accède aux données de manière uniforme, on dit qu'il y a un **faible couplage** (couplage léger ou couplage lâche).
- On parle de **couplage faible**, **couplage léger** ou **couplage lâche** si les composants échangent peu d'information.

Architecture 3-tiers : Couche données

- **Données propres au système :**

- Ces données sont pérennes, car destinées à durer dans le temps, de manière plus ou moins longue, voire définitive.
- Les données peuvent être stockées indifféremment dans de simples fichiers texte, fichiers XML, ou encore dans un SGBD
- Quel que soit le support de stockage choisi, l'accès aux données doit être le même. Cette abstraction améliore la maintenance du système.
- Le DATA ACCESS OBJET (DAO ou design pattern) consiste à représenter les données du système sous la forme d'un modèle objet. Par exemple un objet pourrait représenter un contact ou un rendez-vous.
La représentation du modèle de données objet en base de données (appelée persistance) peut être effectuée à l'aide d'outils tel que HIBERNATE.

Architecture 3-tiers : Couche données

- **Données gérées par un autre système:**
 - Les données peuvent aussi être gérées de manière externe.
 - Elles ne sont pas stockées par le système considéré, il s'appuie sur la capacité d'un autre système à fournir ces informations.
 - Par exemple, une application de pilotage de l'entreprise peut ne pas sauvegarder des données comptables de haut niveau dont elle a besoin, mais les demander à une application de comptabilité.
- Celle-ci est indépendante et pré-existante, et on ne se préoccupe pas de savoir comment elle les obtient ou si elle les sauvegarde, on utilise simplement sa capacité à fournir des données à jour.

Architecture Multi-tiers

- L'architecture multi-tiers est aussi appelée architecture distribuée ou architecture n-tiers.
- L'architecture multi-tiers qualifie la distribution d'applications entre de multiples services et non la multiplication des niveaux de service : les trois niveaux d'abstraction d'une application sont toujours pris en compte.

Architecture Multi-tiers

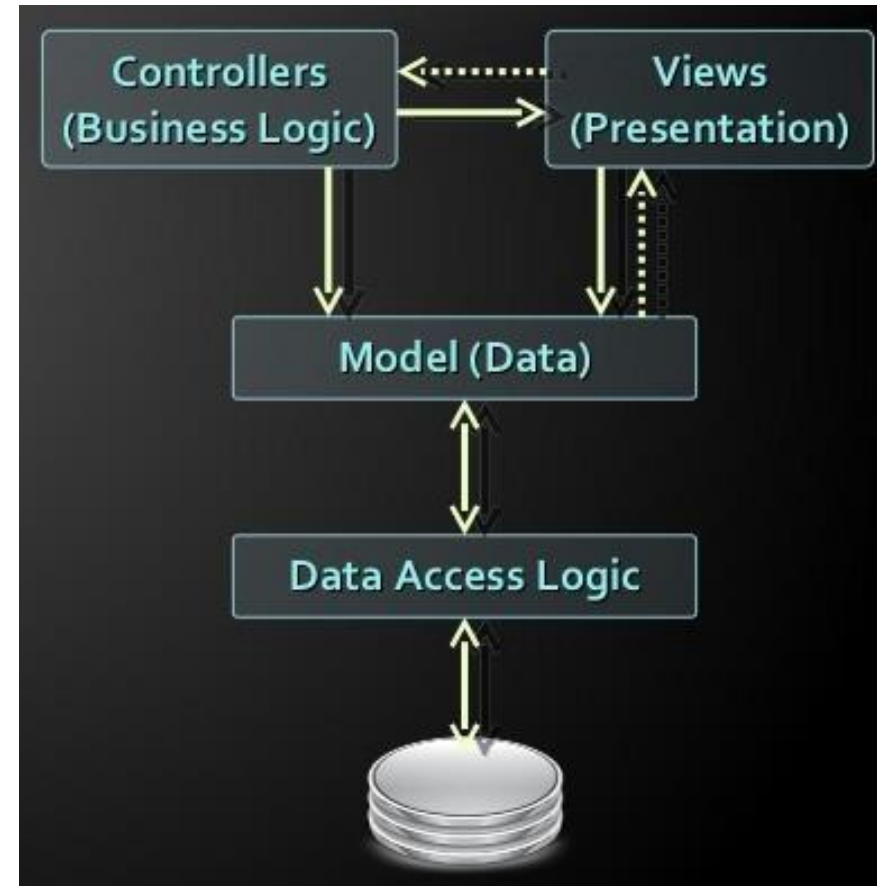
- Cette distribution est facilitée par l'utilisation de **composants métiers**, spécialisés et indépendants, introduits par les concepts orientés objets.
- Elle permet de tirer pleinement partie de la notion de **composants métiers** réutilisables et modulables.
- Un **composant** est une «boîte noire», créé par un développeur, qu'il va réutiliser et que d'autres développeurs vont utiliser. Le développeur utilisateur connaît seulement les points d'entrée et le type des informations retournées. Ex: **Les composants JavaEE sont les EJB.**

Architecture Multi-tiers: Point de vue matériel

- L'architecture présentée en purement logicielle.
- En conséquence, chaque étage peut être implémenté sur n'importe quelle machine.
- Ainsi, Les trois étages peuvent être réunies sur une seule ou sur plusieurs machines.
- L'intérêt de répartir les différents étages sur plusieurs machines permet un accroissement de la sécurité.

Architecture Multi-tiers et Modèle MVC

- Peuvent être utilisées ensemble pour la séparation des couches métiers, données et présentation




Finalemment... Développer un modèle architectural

- Commencer par faire une esquisse de l'architecture
 - En se basant sur les principaux requis des cas d'utilisation ; décomposition en sous-systèmes
 - Déterminer les principaux composants requis
 - Sélectionner un style architectural
- Raffiner l'architecture
 - Identifier les principales interactions entre les composants et les interfaces requises
 - Décider comment chaque donnée et chaque fonctionnalité sera distribuée parmi les différents composants
 - Déterminer si on peut réutiliser un framework existant (réutilisation) ou si on peut en construire un (réutilisabilité).

Développer un modèle architectural avec UML

- Décrire l'architecture avec UML
 - Tous les diagrammes UML peuvent être utiles pour décrire les différents aspects du modèle architectural
 - Trois des diagrammes UML sont particulièrement utiles pour décrire une architecture logicielle
 - Diagramme de packages
 - Diagramme de composants
 - Diagramme de déploiement

Rôle d'un Architecte Logiciel

- Le rôle de l'architecte logiciel est:
 - de définir une architecture logicielle qui satisfasse les contraintes du système
 - de la communiquer et la promouvoir
 - de défendre son intégrité conceptuelle
 - de la critiquer
 - de la raffiner
- 
- A person is seen from behind, looking at a chalkboard covered in various hand-drawn sketches. The sketches include a lightbulb, a pie chart, a bar graph, a speech bubble with the word 'SUCCESS', a cloud with a rainbow, a dollar sign, and the word 'IDEA'. There are also some numbers like '100,000.0' and '100,000.' and various arrows and lines connecting the different elements.



Profil d'un Architecte Logiciel

- Crédibilité
- Pensée intégrative
- Créativité
- Empatie
- Autodidacte
- Leader/Mentor
- Communicateur(Négociateur)
- Excellent analyste
- Designer hors pair
- Bon programmeur



7 Périls d'un Architecte Logiciel

- Un projet où la direction de l'organisation ne croie pas à l'architecture logicielle
- Un projet dont les usagers ne veulent pas
- Embarquer sur un projet sans avoir de crédibilité auprès d'une équipe rebelle
- Un projet avec un niveau d'incertitude élevé soumis à un développement en cascade
- Un projet dont des choix technologiques clés imposés sont inappropriés
- Un projet où les analystes d'affaires produisent le schéma de BD
- Prendre la relève d'un projet en détresse

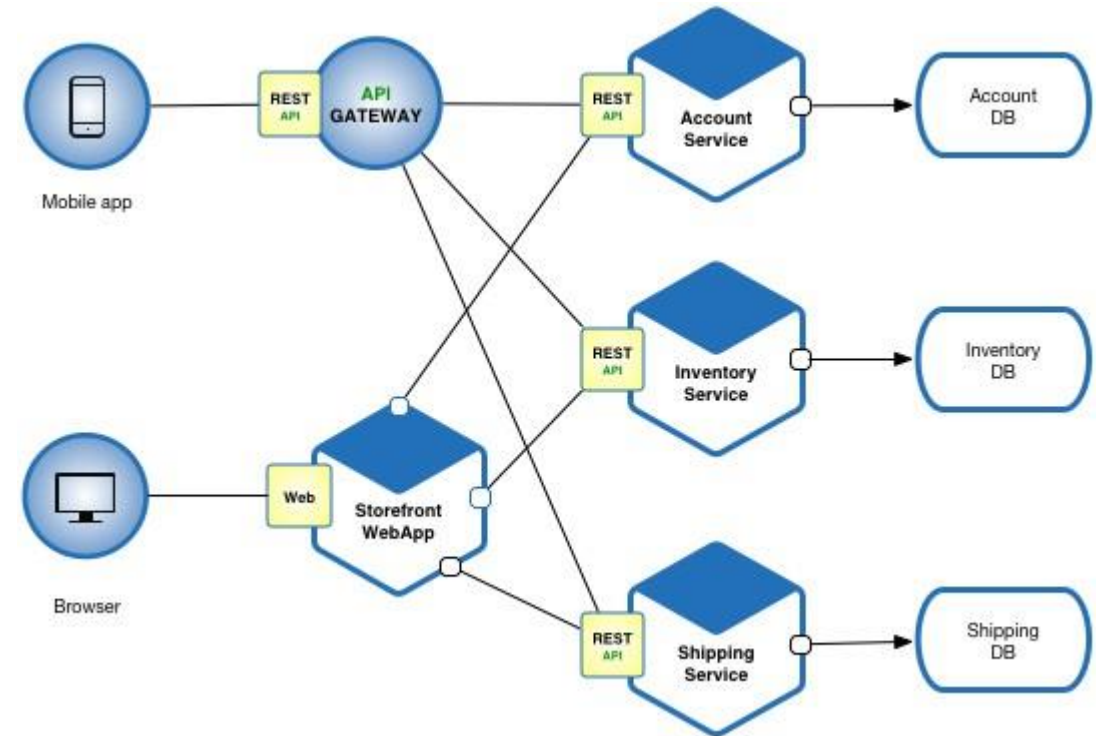
Pour terminer la tendance: Architecture Microservices

- En [informatique](#), les microservices sont un [style d'architecture logicielle](#) à partir duquel un ensemble complexe d'[applications](#) est décomposé en plusieurs processus indépendants et faiblement couplés, souvent spécialisés dans une seule tâche. Les [processus indépendants](#) communiquent les uns avec les autres en utilisant des [API](#) langage-agnostiques.
- Des API [REST](#) sont souvent employées pour relier chaque microservice aux autres. Un avantage avancé est que lors d'un besoin critique en une ressource, seul le microservice lié sera augmenté, contrairement à la totalité de [l'application dans une architecture classique, par exemple une architecture trois tiers](#). Cependant, le coût de mise en place, en raison des compétences requises, est parfois plus élevé. <https://fr.wikipedia.org/wiki/Microservices>

<https://www.supinfo.com/articles/single/5676-qu-est-ce-que-architecture-microservices>
pour plus de détails voir ce lien.

Pour terminer la tendance: Architecture Microservices

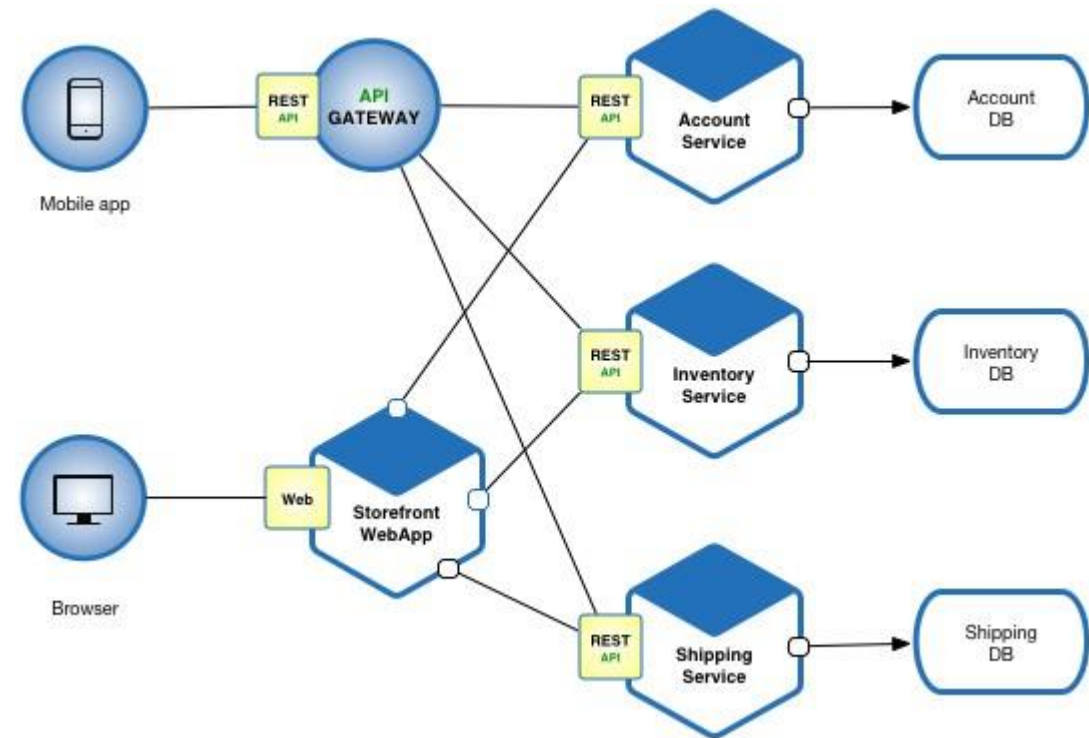
Imaginons que vous construisiez une application de commerce électronique qui prend les commandes des clients, vérifie l'inventaire et le crédit disponible, et les expédie. L'application se compose de plusieurs composants, y compris le StoreFrontUI, qui implémente l'interface utilisateur, ainsi que certains services de backend pour vérifier le crédit, maintenir l'inventaire et les commandes d'expédition. L'application consiste en un ensemble de services.



Architecture Microservices

Imaginons que vous construisiez une application de commerce électronique qui prend les commandes des clients, vérifie l'inventaire et le crédit disponible, et les expédie. L'application se compose de plusieurs composants, y compris le StoreFrontUI, qui implémente l'interface utilisateur, ainsi que certains services de backend pour vérifier le crédit, maintenir l'inventaire et les commandes d'expédition. L'application consiste en un ensemble de services.

[Amazon](#), [Bluemix](#), [Cloud Foundry](#), [Google](#), [The Guardian](#), [Hailo Taxi](#), [HP Helion Development Platform](#), [Jelastic](#), [Devslack](#), [Microsoft Azure](#), [Netflix⁵](#) (Netflix reçoit 2 milliards de requêtes chaque jour, conduisant à environ 20 milliards d'appels à des API internes), [Riot Games](#), [SoundCloud](#), [Uber](#) et d'autres sites Web à grande échelle et les applications ont tous évolué de l'architecture monolithique à microservices.



Biblio et Webographie

- Len Bass, Paul Clements, and Rick Kazman(2003).Software architecture in practice,2nd edition,Addison-Wesley.
- David Garlan. 2000. Software architecture: a roadmap. In Proceedings of the Conference on The Future of Software Engineering (ICSE '00). ACM, New York, NY, USA, 91-101. DOI=<http://dx.doi.org/10.1145/336512.336537>
- Cours de Yann-Gaël Guéhéneuc au Département de génie informatique et logiciel de l'École Polytechnique de Montréal <http://www.yann-gael.gueheneuc.net/Work/Teaching/>
- <https://apiumhub.com/tech-blog-barcelona/benefits-of-software-architecture/>
- <https://www.supinfo.com/articles/single/5676-qu-est-ce-que-architecture-microservices>
- <https://techbeacon.com/top-5-software-architecture-patterns-how-make-right-choice>