Cours 2
Introduction à l’architecture Web

Un cours de Yann Fornier

Plan

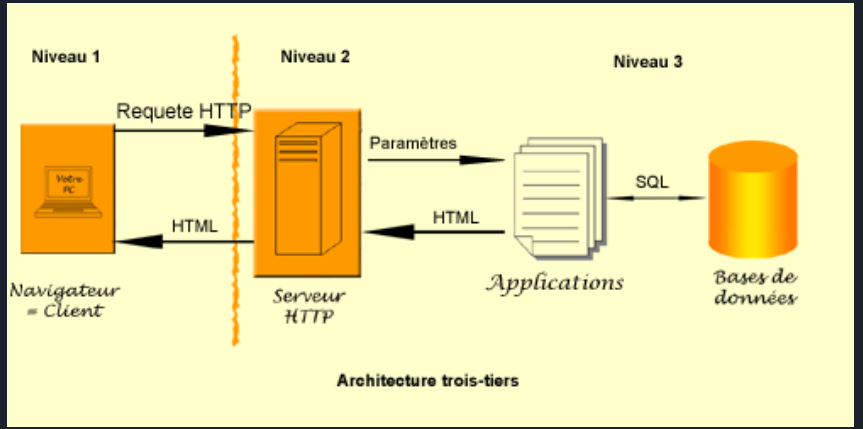
Session 2 : Introduction à l’architecture web

- Définition de l’architecture web et des différents types d’architecture (monolithe, microservices, etc.)
- Les avantages et inconvénients de chaque type d’architecture
- **Exercice** : modéliser une architecture web en utilisant un diagramme de cas d’utilisation

Qu’est ce que l’architecture web ?

L’architecture web désigne la manière dont les différents éléments d’une application web (services, bases de données, front-end, etc.) sont organisés et interagissent entre eux.

Elle détermine la façon dont l’application web sera conçue, développée, déployée et maintenue.



Qu’est ce que la scalabilité ?

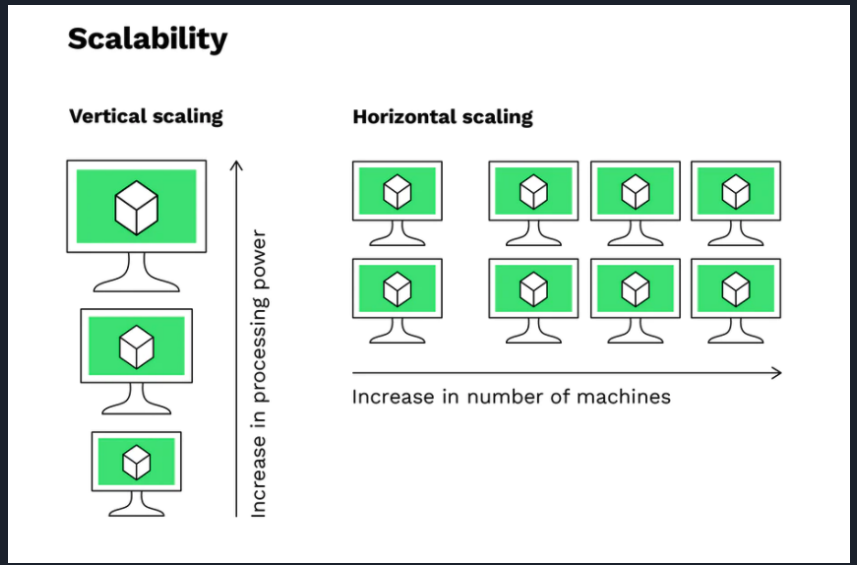
La scalabilité est la capacité d’un système à s’adapter et à évoluer en fonction de la croissance de la demande en termes de ressources, de performance et de fonctionnalités.

Qu’est ce que la scalabilité ?

Un exemple de scalabilité serait un site web qui peut gérer une forte augmentation du nombre de visiteurs sans ralentir ou tomber en panne.

Pour atteindre cette scalabilité, le site web peut utiliser des techniques de load balancing pour répartir la charge de travail sur plusieurs serveurs, ou bien utiliser une architecture microservices pour permettre à chaque fonctionnalité de l’application de s’évoluer indépendamment des autres.

Il est important de prendre en compte la scalabilité lors de la conception d’une application, car elle peut avoir un impact significatif sur sa performance et sa fiabilité à long terme.



Différentes architectures

Il existe plusieurs types d'architecture web, qui varient selon la manière dont les différents éléments de l'application sont séparés et communiquent entre eux.

Architecture Monolithe

Architecture Microservices

Architecture Serverless

Il n'y a pas de type d'architecture web qui convient à tous les cas d'utilisation, et le choix de l'architecture dépend de nombreux facteurs tels que la taille de l'application, les contraintes de temps et de budget, les exigences en matière de performances et de disponibilité, etc.

Travail de groupe noté

Présentation des architectures par groupe et de leurs enjeux. (Powerpoint demandé)

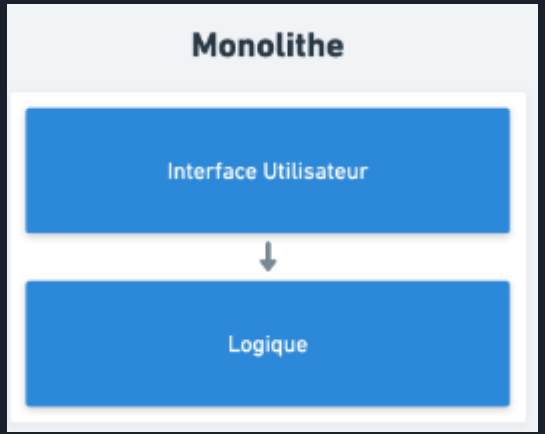
Durée : 30 minutes

Groupe 1	Groupe 2	Groupe 3
Architecture Monolithe	Architecture Microservices	Architecture Serverless

Architecture monolithe

L’architecture monolithe est une approche de développement d’application web qui consiste à regrouper tous les éléments de l’application dans un seul et même codebase et à les déployer ensemble.

Elle est souvent utilisée pour les applications de petite et moyenne taille, qui n’ont pas besoin de scalabilité ou de flexibilité élevées.



Avantages de l’architecture monolithe

Simplicité de mise en œuvre et de déploiement : l’architecture monolithe est très simple à mettre en place et à déployer, car elle ne nécessite pas de gérer de nombreux microservices ou fonctions indépendants. Il suffit de développer l’application dans un seul codebase et de la déployer en une seule fois sur un serveur ou une plateforme de cloud.

Avantages de l’architecture monolithe

Facilité de développement et de maintenance : dans l’architecture monolithe, tous les éléments de l’application sont regroupés dans le même codebase, ce qui rend le développement et la maintenance plus faciles. Les développeurs peuvent travailler sur tous les aspects de l’application en même temps et utiliser les mêmes outils de développement et de versionning.

Avantages de l’architecture monolithe

Coûts de développement et de maintenance réduits : l’architecture monolithe permet de réduire les coûts de développement et de maintenance, car elle ne nécessite pas de gérer de nombreux microservices ou fonctions indépendantes. Elle est donc particulièrement adaptée aux petites et moyennes applications qui n’ont pas besoin de scalabilité ou de flexibilité élevées.

Inconvénients de l’architecture monolithe

Manque de scalabilité et de flexibilité : l’architecture monolithe ne permet pas de scaler facilement les différents éléments de l’application de manière indépendante. Si une partie de l’application devient très populaire et nécessite plus de ressources, il faut déployer une nouvelle version de l’application entière, ce qui peut être coûteux et prendre du temps. De même, il est difficile de mettre à jour ou de remplacer une partie de l’application sans affecter l’ensemble de l’application.

Inconvénients de l’architecture monolithe

Complexité croissante à mesure que l’application grandit : l’architecture monolithe peut devenir de plus en plus complexe à mesure que l’application grandit et qu’elle inclut de nouvelles fonctionnalités. Il peut être difficile de maintenir une vue d’ensemble de l’application et de la développer de manière cohérente, ce qui peut entraîner des problèmes de qualité et de performance.

Inconvénients de l’architecture monolithe

Difficulté à gérer les dépendances entre les éléments de l’application : dans l’architecture monolithe, il est difficile de gérer les dépendances entre les différents éléments de l’application, ce qui peut entraîner des problèmes de développement et de maintenance.

Exemples d’applications monolithes

Un site de e-commerce de petite et moyenne taille, qui propose une gamme limitée de produits et qui n’a pas besoin de scalabilité élevée.

Un site de blog ou de journal en ligne, qui affiche du contenu statique et qui n’a pas besoin de fonctionnalités de mise à jour en temps réel.

Un site de gestion de projet pour une équipe restreinte, qui ne nécessite pas de scalabilité élevée ni de flexibilité.

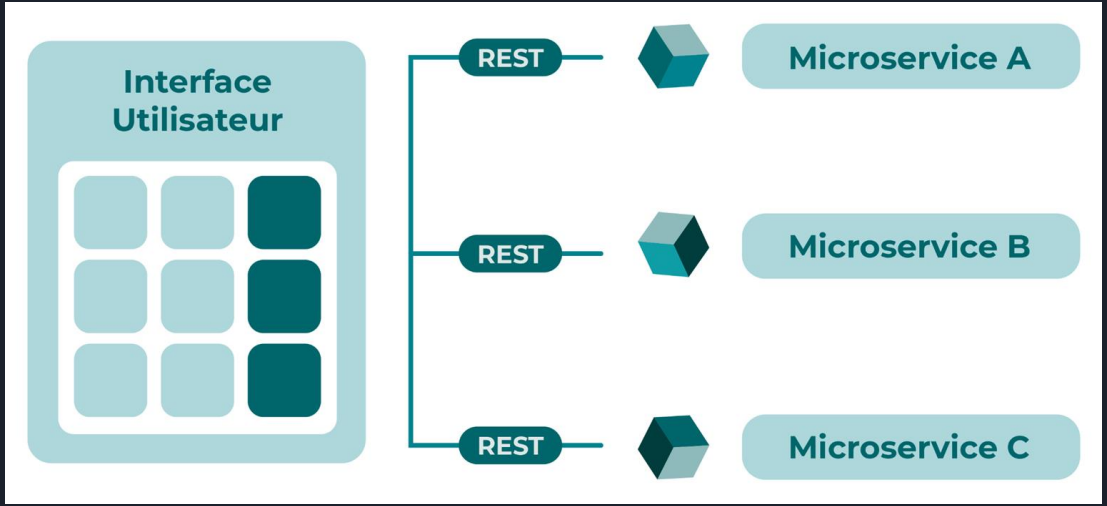
Conclusion

L’architecture monolithe est une approche de développement d’application web simple et adaptée aux applications de petite et moyenne taille qui n’ont pas besoin de scalabilité ou de flexibilité élevées.

Elle présente des avantages en termes de simplicité de mise en œuvre et de déploiement, de facilité de développement et de maintenance, et de coûts réduits. Cependant, elle peut être moins adaptée aux applications qui ont besoin de scalabilité ou de flexibilité élevées, ou qui sont amenées à grandir de manière importante, car elle peut devenir complexe et difficile à maintenir dans ces cas.

L’architecture microservices

L’architecture microservices est une approche de développement d’application web qui consiste à regrouper les éléments de l’application en plusieurs microservices indépendants, qui communiquent entre eux pour réaliser les fonctionnalités de l’application. Cette approche est souvent utilisée pour les applications de grande taille ou à forte croissance, qui ont besoin de scalabilité et de flexibilité élevées.



Avantages de l’architecture microservices

Scalabilité et flexibilité élevées : dans l’architecture microservices, chaque microservice est indépendant et peut être déployé et évolué de manière indépendante. Cela permet de scaler facilement les différents éléments de l’application de manière indépendante et de mettre à jour ou de remplacer une partie de l’application sans affecter l’ensemble de l’application.

Avantages de l’architecture microservices

Facilité de développement et de maintenance : dans l’architecture microservices, chaque microservice est développé et maintenu par une équipe dédiée, ce qui rend le développement et la maintenance plus faciles. Les développeurs peuvent travailler sur un microservice en particulier et utiliser les outils de développement et de versionning qui leur conviennent le mieux.

Avantages de l’architecture microservices

Meilleure gestion des dépendances entre les éléments de l’application : dans l’architecture microservices, chaque microservice est indépendant et n’a pas de dépendances fortes avec les autres microservices. Cela permet de mieux gérer les dépendances entre les différents éléments de l’application et de limiter les problèmes de développement et de maintenance.

Inconvénients de l’architecture microservices

Coûts de développement et de maintenance élevés : l’architecture microservices nécessite de gérer de nombreux microservices indépendants, ce qui peut augmenter les coûts de développement et de maintenance. Il faut également prévoir des outils de gestion de microservices, comme un registry ou un service mesh, ce qui peut ajouter des coûts supplémentaires.

Inconvénients de l’architecture microservices

Complexité croissante à mesure que l’application grandit : l’architecture microservices peut devenir de plus en plus complexe à mesure que l’application grandit et qu’elle inclut de nouveaux microservices. Il peut être difficile de maintenir une vue d’ensemble de l’ensemble des microservices et de leurs interactions, ce qui peut entraîner des problèmes de qualité et de performance.

Inconvénients de l’architecture microservices

Difficulté à gérer les erreurs et les pannes : dans l’architecture microservices, il est difficile de gérer les erreurs et les pannes, car chaque microservice est indépendant et peut être impacté par des problèmes de disponibilité ou de performance. Il faut mettre en place des outils de monitoring et de gestion des erreurs pour surveiller l’état des microservices et intervenir rapidement en cas de problème.

Exemples d’applications web microservices

Un site de e-commerce de grande taille, qui propose une large gamme de produits et qui a besoin de scalabilité élevée pour gérer de nombreux utilisateurs et commandes.

Un site de médias en ligne, qui affiche du contenu dynamique et qui a besoin de fonctionnalités de mise à jour en temps réel.

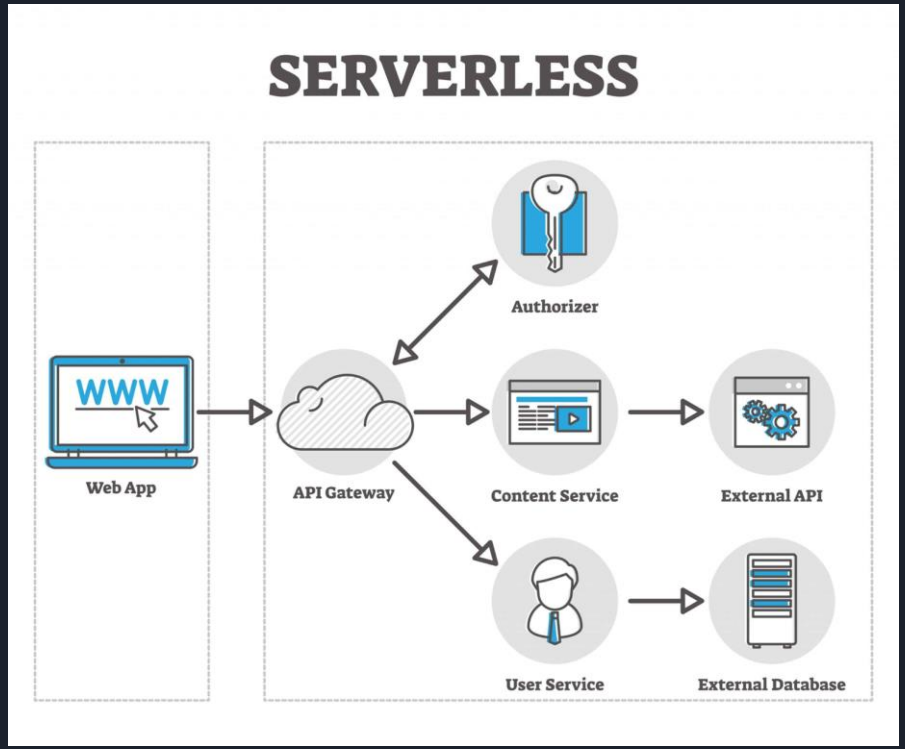
Un site de gestion de projet pour une équipe de grande taille, qui nécessite une flexibilité élevée et la possibilité de développer et de mettre à jour rapidement les différentes fonctionnalités.

Conclusion

L’architecture microservices est une approche de développement d’application web adaptée aux applications de grande taille ou à forte croissance qui ont besoin de scalabilité et de flexibilité élevées. Elle présente des avantages en termes de scalabilité, de flexibilité, de facilité de développement et de maintenance, et de meilleure gestion des dépendances.

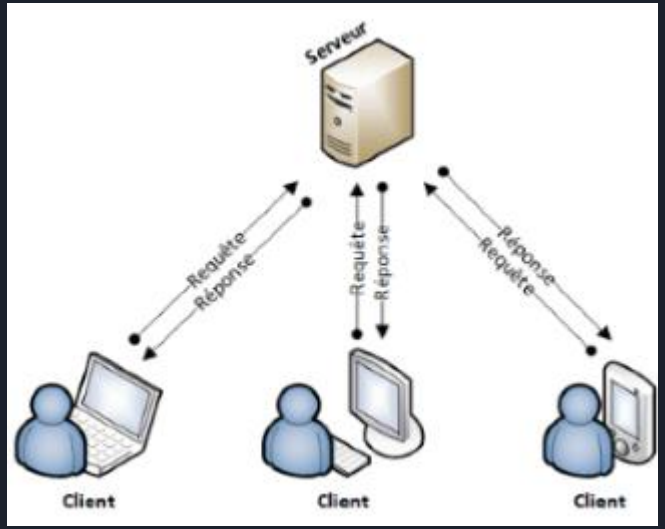
Cependant, elle peut être moins adaptée aux applications de petite ou moyenne taille ou à faible croissance, car elle peut être coûteuse et complexe à mettre en œuvre et à maintenir. Il est important de bien évaluer les exigences en termes de scalabilité, de flexibilité, de coûts et de délai, afin de choisir l’architecture la plus adaptée à votre projet.

L’architecture Serverless



Acteurs et protocoles de communication en architecture web

Dans une architecture web, il y a plusieurs acteurs qui jouent un rôle clé dans la communication et l'interaction entre les différents éléments de l'application. Ces acteurs sont le client, le serveur et le réseau.



Le Client

Le client est l’application ou le dispositif qui envoie une demande de données ou de services à un serveur. En architecture web, le client peut être un navigateur web (comme Chrome, Firefox ou Safari), une application mobile (comme une application Android ou iOS), ou tout autre dispositif qui peut envoyer des requêtes HTTP (Hypertext Transfer Protocol) à un serveur.

Le serveur

Le serveur est l'application ou le dispositif qui reçoit les demandes de données ou de services envoyées par le client, et qui envoie une réponse en retour. En architecture web, le serveur peut être un serveur web (comme Apache ou Nginx), un serveur d'application (comme Tomcat ou Wildfly), ou tout autre dispositif qui peut recevoir des requêtes HTTP et envoyer des réponses.

Serveur Web

Serveur d'application

Serveur de BDD

Le serveur Web

Un serveur web est un logiciel qui exécute sur un ordinateur et qui est conçu pour accepter des requêtes HTTP (Hypertext Transfer Protocol) en provenance de clients, tels que des navigateurs web, et pour renvoyer des réponses HTTP à ces clients. Les serveurs web sont souvent utilisés pour héberger des sites web, c'est-à-dire pour stocker, traiter et fournir les fichiers qui composent les pages web aux utilisateurs sur Internet.

Le serveur Web

Les serveurs web sont généralement exécutés sur des ordinateurs qui sont connectés à Internet en permanence et qui sont configurés pour accepter des connexions réseau entrantes. Lorsqu'un client envoie une requête HTTP à un serveur web, le serveur analyse la requête et envoie une réponse appropriée au client, qui peut être une page web, une image, un fichier audio ou tout autre type de données.

Le serveur Web

Il existe plusieurs types de serveurs web, y compris les serveurs web Apache, Nginx et Microsoft IIS, qui sont tous largement utilisés sur Internet. En général, les serveurs web sont configurés et gérés par des administrateurs système ou des développeurs web, qui s’assurent que le serveur fonctionne correctement et de manière sécurisée.

Le serveur d’application

Un serveur d’application est un logiciel qui exécute sur un ordinateur et qui est conçu pour gérer les applications et les services qui sont accessibles sur un réseau. Les serveurs d’application sont généralement utilisés pour fournir une interface de programmation (API) qui peut être utilisée par d’autres applications ou par des utilisateurs pour accéder à ces services.

Le serveur d’application

Les serveurs d’application sont souvent utilisés pour exécuter des applications de gestion de bases de données, des applications de gestion de contenu, des applications de commerce électronique et de nombreuses autres applications qui nécessitent une interface de programmation pour être accessibles par d’autres logiciels ou par des utilisateurs finaux. Ils peuvent également être utilisés pour fournir des services tels que l’authentification, l’autorisation et la gestion des utilisateurs à d’autres applications.

Le serveur d’application

Il existe plusieurs types de serveurs d’application, tels que les serveurs d’application Java, les serveurs d’application Microsoft .NET et les serveurs d’application Ruby on Rails, Python Django … En général, les serveurs d’application sont configurés et gérés par des administrateurs système, SRE, Devops ou des développeurs d’application, qui s’assurent que le serveur fonctionne correctement et de manière sécurisée.

Serveur de base de données

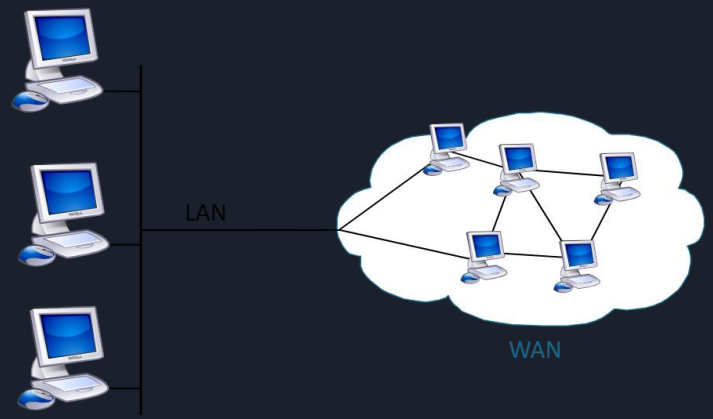
Les serveurs de base de données sont souvent utilisés pour stocker et gérer les données de grandes organisations, telles que les entreprises, les gouvernements et les universités. Ils offrent une interface de programmation (API) qui peut être utilisée par d'autres applications ou par des utilisateurs pour accéder et gérer les données dans la base de données.

Serveur de base de données

Il existe plusieurs types de serveurs de base de données, tels que MySQL, Microsoft SQL Server et Oracle, qui sont tous largement utilisés dans de nombreux domaines. En général, les serveurs de base de données sont configurés et gérés par des administrateurs de base de données, qui s'assurent que le serveur fonctionne correctement et de manière sécurisée.

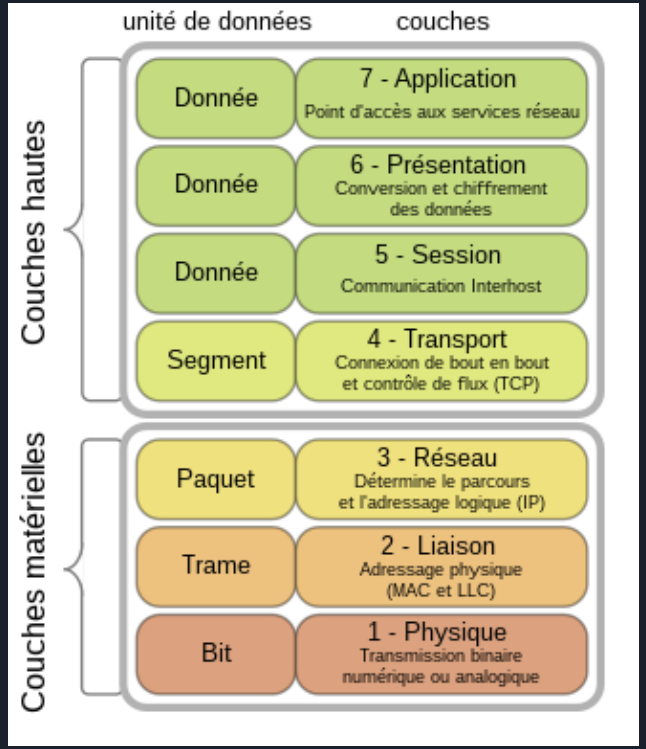
Le réseau

Le réseau est l’ensemble des équipements et des protocoles qui permettent la communication entre le client et le serveur. En architecture web, le réseau peut être le réseau local (LAN), le réseau étendu (WAN), ou Internet.



Les protocoles de communication

Un protocole de communication est un ensemble de règles et de conventions qui définissent comment deux ou plusieurs parties (par exemple, des ordinateurs, des serveurs, des appareils mobiles) peuvent échanger des informations sur un réseau (par exemple, Internet, un LAN). Il permet de standardiser la manière de formater, de transmettre et de recevoir les données, afin de garantir une compréhension mutuelle et une interaction fiable entre les parties.

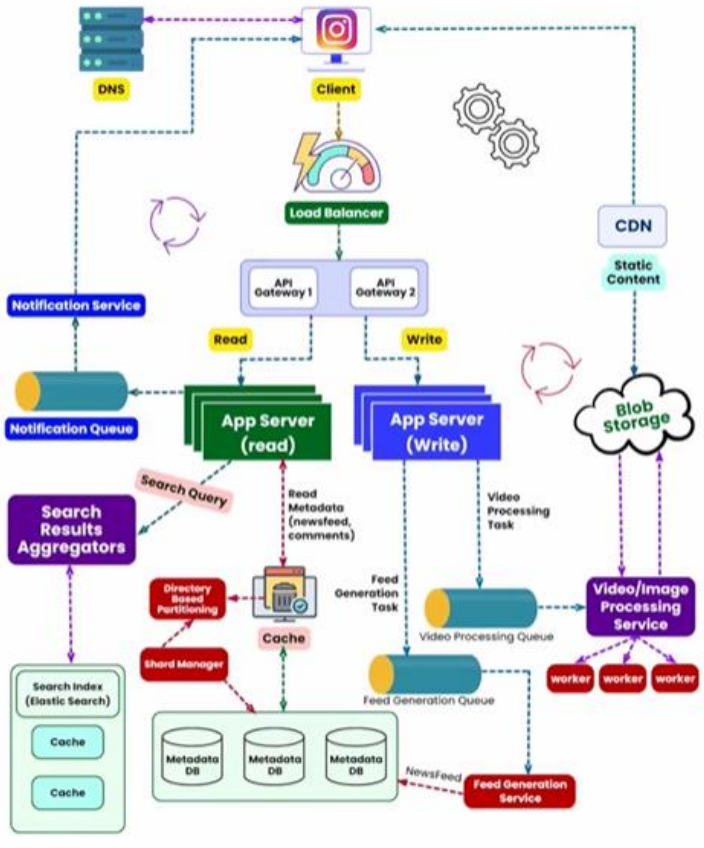


Exercice : Modéliser une architecture web en utilisant un diagramme de microservices

Vous pouvez utiliser un outil de diagramme de microservices comme Lucidchart ou Draw.io pour réaliser votre modélisation. Voici quelques éléments à prendre en compte lors de la création de votre diagramme :

Votre cas d'étude sera le réseau instagram, pour faciliter l'exercice on exclut volontairement la partie "chat" et "story" de l'application.

Instagram System Design



@learnwithrockybhatia