Étude de l’architecture logicielle de la plateforme YouTube

Mars 2023

Alexandre Huynh
Varun Vedic Gupta

Table des matières

Introduction sur la plateforme YouTube

Nous avons choisi de traiter l'architecture logicielle de YouTube, plateforme de diffusion de vidéos en ligne la plus utilisée dans le monde depuis sa création en 2005 par les trois fondateurs Steve Chen, Chad Hurley et Jawed Karim [1]. Avec une estimation de 2,6 milliards d'utilisateurs actifs mensuels en 2023 [2], la plateforme doit s'adapter à la charge en performance liée au stockage et la diffusion de vidéos, afin de fournir ses services rapidement et sans erreurs, tout en préservant la sécurité des données et de ses utilisateurs. Pour cela, la plateforme a donc débuté le développement de son application en suivant une architecture en couches afin d'assurer les bases de son fonctionnement [3] [4], puis suivant les évolutions de ses services proposés, YouTube a migré progressivement certains services vers une architecture micro-services pour s'adapter aux divers changements [4] [5].

L'architecture en couches

1. Présentation de l'architecture en couches

Avant de s'intéresser à l'application de l'architecture en couches, rappelons le concept même de celle-ci. Elle est généralement composée de 5 couches et se base sur les différents aspects de l'application [6] [7].

Commençons par la couche la plus visible de l'application, la couche d'interaction avec l'utilisateur (User Interaction layer) permet à l'utilisateur d'interagir avec un système par le biais d'une interface utilisateur. Concrètement, l'UIL inclut les différents éléments qui permettent à l'utilisateur de naviguer sur le système facilement. Ces éléments peuvent prendre la forme d'interfaces graphiques, de boutons, de formulaires. En effet, ces différents éléments qui prennent en compte les différents interactions de l'utilisateur et du système (comme par exemple la saisie de texte de l'utilisateur ou d'autres interactions qui nécessitent l'action de l'utilisateur mais également les données exposées à ce dernier à travers différents graphes et tableaux) qui se doivent d'être claires pour une expérience utilisateur agréable et optimale [6] [7].

La couche située en dessous de la couche interaction avec l'utilisateur est la couche de fonctionnalités (Functionality Layer) et au-dessus de la couche data layer. La couche fonctionnalité appelée aussi "la couche logique métier" est l'ensemble des opérations qui vont définir les différents comportements et fonctionnalités d'une application [6] [7]. Elle établit un lien entre l'interface avec ses boutons et éléments, avec les fonctionnalités de l'application qui se traduisent par des lancements de fonctions, des traitements.

La couche située encore en dessous est la couche des règles de gestion (Business Rules layer), qui définit le comportement de l'ensemble de l'application. Concrètement, cela permet de structurer les différentes utilisations possibles en maintenant la sécurité de l'application, en assurant la cohérence et l'interopérabilité entre les différentes couches, tout en garantissant que les données sont valides [7]. Cette couche contient parfois des algorithmes se basant sur les actions de l'utilisateur ou des données fournies, qui vont fonctionner d'une certaine manière et appelleront des fonctions spécifiques.

La couche Noyau de l'application (Application Core) désigne les principaux programmes de l'application avec les diverses fonctions de base de l'application [6] [7]. Tous les traitements des fonctionnalités de l'application sont contenus dans cette couche à travers

leur code. Les fonctions de cette couche interagissent avec la base de données en se basant sur les règles de gestion, et sont sollicitées quand un utilisateur clique par exemple sur un bouton depuis l'interface.

Enfin, la couche de la base de données (Database layer) concerne la base de données comme évoqué : elle contient les tables, index et données de l'application [6] [7], et supporte toute exécution d'opérations d'insertion, de modification ou suppression de données.

2. Exemples d'entreprises

Certaines grandes entreprises proposent des applications complexes avec de nombreux services et fonctionnalités, employant l'architecture en couches afin de garantir sa scalabilité, c'est-à-dire sa capacité à évoluer avec par exemple de nouveaux services, tout en maintenant un fonctionnement correct et adapté aux niveaux performance et sécurité face à un grand nombre de données et d'utilisateurs. Nous pouvons prendre en exemple Oracle avec ses applications Java [8], ou encore Microsoft qui fait usage de cette architecture pour ses applications business tel que Microsoft 365 [8].

3. Les débuts de YouTube avec l'architecture en couches

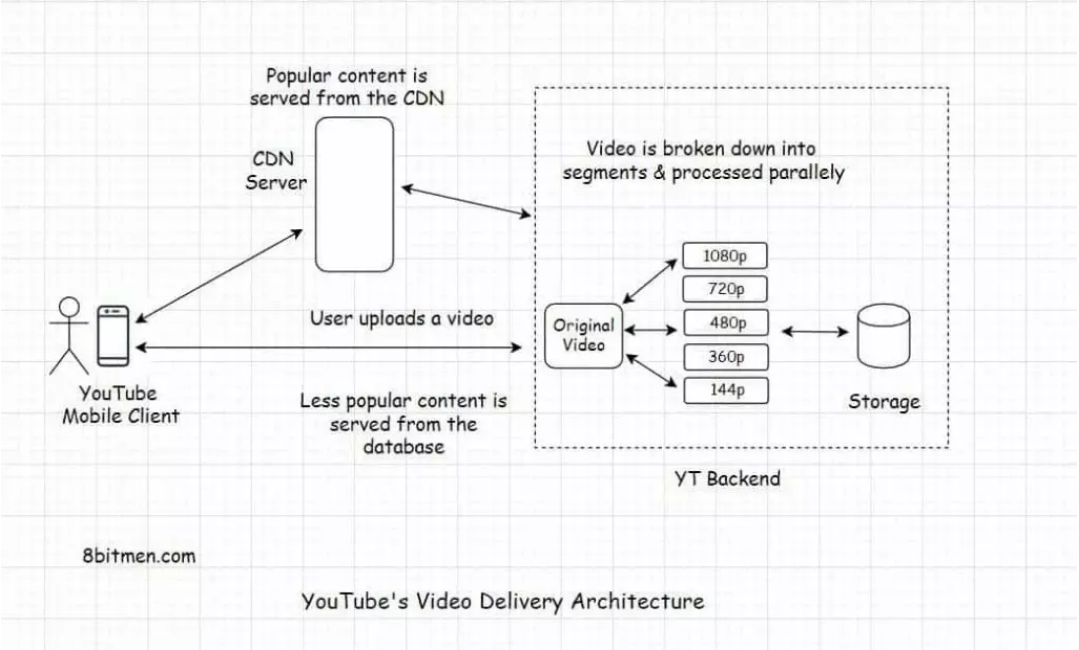
La plateforme YouTube étant très complète en fonctionnalités, elle a été conçue dans un premier temps avec une architecture en couches afin de gérer la complexité de sa structure et de permettre divers évolutions dans son utilisation ou à travers de nouvelles fonctionnalités. Nous pouvons distinguer les couches mentionnées précédemment.

Premièrement, YouTube apporte une attention à la couche d'interaction avec l'utilisateur, notamment pour définir l'interface graphique de l'application, la présentation et la redirection vers les vidéos, et l'accès aux différentes fonctionnalités. L'aspect Responsive Design est aussi pris en compte, la plateforme étant accessible sur tous types d'appareils, ordinateur, tablette ou encore smartphone. Les langages HTML, CSS et JavaScript sont mobilisés pour assurer la partie visuelle de l'application.
En suivant l'architecture en couches, YouTube peut apporter des ajouts et mises à jour fréquentes de l'apparence visuelle du site sans impacter le fonctionnement de l'application en elle-même.

Deuxièmement, la plateforme met en place à travers du code JavaScript un grand nombre de fonctionnalités en lien avec l'interface de l'application, comme pour le lecteur de vidéos ou le système de commentaire. En prenant exemple du lecteur vidéo, le JavaScript permet d'interagir avec la lecture de la vidéo à travers divers boutons pour lancer la vidéo, la mettre en pause, augmenter ou réduire le volume audio, modifier la qualité audio, et d'autres encore. Nous pouvons aussi évoquer la barre de recherche, élément essentiel dans l'utilisation de la plateforme de diffusion, répertoriant des milliards de vidéos en ligne. Ceux-ci n'impactent pas directement le système, mais ils créent un lien entre l'interface visuelle (la couche d'interaction avec l'utilisateur) et les traitements back-end (les couches des règles de gestion et de noyau d'application) qui vont être déclenchés selon l'utilisation de YouTube par les utilisateurs, à travers un clic par exemple.

Troisièmement, YouTube fait usage d'un grand nombre de règles de gestion variées sous forme d'algorithmes pour un fonctionnement efficace et productif de la plateforme. En effet, un des principes les plus importants porte sur le système de recommandation personnalisée des vidéos, qui se réfère à l'historique de visionnage des utilisateurs [4]. Elle implique l'utilisation de machine learning qui, en se basant sur ces facteurs et plusieurs algorithmes spécifiques, influence donc quelles vidéos vont être proposées à l'utilisateur à travers les différentes pages de l'application, tout en mobilisant les diverses autres couches.
En prenant en compte l'architecture en couches, les règles de gestion concernant les recommandations vont avoir un impact sur les fonctions qui seront appelées pour les services de l'application : on peut supposer qu'elle décidera de quels types de vidéo à proposer, quels sujets en particulier, et cherchera donc le contenu média adapté dans la base de données.

Quatrièmement, la couche de noyau de l'application YouTube comprend donc une très grande partie des traitements back-end en relation avec les bases de données du groupe, servant à l'implémentation de diverses fonctionnalités telles que la publication de vidéos, leur recommandation, l'analyse de statistiques et la distribution de contenu média. En effet, la simple lecture d'une vidéo par un utilisateur exécute un grand nombre de code Python [9], incluant notamment des traitements via les algorithmes de recommandations et un système de machine-learning mentionnés précédemment se basant sur l'historique des utilisateurs pour des suggestions précises et personnalisées [4].
Nous pouvons aussi noter les traitements effectués lors de la publication d'une vidéo par un utilisateur, passant par deux étapes majeures : le téléversement du contenu vidéo et la mise à jour des métadonnées [10].
D'une part, les vidéos sont encodées et transcodées dans plusieurs formats et résolutions lors de la mise en ligne de vidéos, afin d'être diffusable sur différents appareils. Celle-ci se fait à travers l'utilisation de codecs tels que H264 et avec des langages comme Python ou C++, et est envoyé par la suite vers la base de données et commence la mise à jour des métadonnées de la vidéo envoyée [10].
D'autre part, le client utilisateur qui a publié la vidéo envoie une requête à la base de données afin de mettre à jour les métadonnées, concernant diverses informations telles que le titre de la vidéo, son URL, la miniature utilisée, les informations de l'utilisateur et d'autres encore [10].
Ceci étant dit, l'emploi de l'architecture en couches permet à YouTube d'implémenter les traitements fondamentaux à l'usage de son application, ici donc la mise en ligne de vidéos par des utilisateurs.



Source illustration : Article [“YouTube Architecture – How Does It Serve High-Quality Videos With Low Latency”] sur le blog scaleyourapp.com

Cinquièmement, la plateforme mobilise une quantité très importante de données dû au stockage des milliards de vidéos sur les serveurs de la société, qui dans la presque-totalité des cas occupent chacun une taille non-négligeable. Le stockage de ces derniers a débuté avec des bases de données MySQL installées sur plusieurs serveurs, qui a ensuite évolué vers une répartition à travers plusieurs data-center, des centres de données réparties dans le monde entier sur plusieurs localisations. En effet, compte tenu de l’évolution massive du volume d’informations et du nombre d’utilisateurs sollicitant les bases de données, YouTube utilise depuis 2012 le framework Vitess en plus de MySQL afin d’optimiser les performances des bases de données [5]. Ce framework a été développé par YouTube lui-même en 2010 pour des usages internes [11], et permet à la plateforme de partitionner les bases de données en échelle horizontale, c’est-à-dire de diviser une grande base de données en plusieurs partitions, autrement connus sous le terme anglais de “shards”. Cette méthode permet donc aussi de les distribuer à plusieurs serveurs, notamment avec le réseau Media CDN (Content Delivery Network) des services Google Cloud Storage. Ces derniers permettent donc de répliquer et stocker les vidéos dans plusieurs centres afin de permettre un accès rapide au contenu. En effet, quand un client demande une vidéo populaire, il se connecte au CDN le plus proche de sa localisation afin de réduire considérablement la latence. La stratégie de YouTube concernant le stockage des vidéos consiste à diffuser les contenus populaires depuis les serveurs CDN réparties dans le monde, tandis que les contenus moins populaires sont fournis depuis les serveurs de bases de données [3].

Avec l’architecture en couches, la plateforme bénéficie d’une base de données extensible et évolutive, tout en étant séparée des traitements fonctionnels. Elle est donc purement dédiée seulement au stockage des données (ici, les vidéos et les métadonnées) qui seront mobilisés à travers des requêtes par la couche noyau de l’application, concernant donc les traitements back-end.

Ceci étant dit, YouTube possède donc une base fondamentale pour le fonctionnement de son application, avec une séparation en plusieurs couches favorisant les interactions entre chacune et permettant de distinguer les différents composants. Cependant, sa force se définit aussi comme une faiblesse, car ces couches sont donc aussi dépendantes entre elles : si un des composants est défaillant, le fonctionnement entier de l'application peut être impacté, et la résolution du problème nécessite une réévaluation du système qui peut s'avérer complexe et onéreux en temps et en argent [6]. YouTube s'est donc progressivement orienté vers une nouvelle architecture : l'architecture microservices.

L'architecture microservices

1. Présentation de l'architecture en micro-services

L'architecture microservices est une architecture dérivée de celle orienté services (Service Oriented Architecture) [14] qui a été adoptée par de nombreuses grandes entreprises pour faire face à la complexité des applications volumineuses, en les décomposant en un ensemble de services indépendants communiquant entre eux via des API, tout en ayant une base de données propre à chacun des services [12] [14]. Cela répond aux difficultés liés aux mises à jour et à la maintenance des couches évoqués précédemment, étant donné qu'ils sont indépendants [14]. Tout changement se fait plus facilement sans se préoccuper de la dépendance entre services qui pourrait ralentir ou bloquer la progression [12]. Cependant, elle présente aussi des défauts, notamment concernant la complexité du système, du développement et des tests [12] [14].

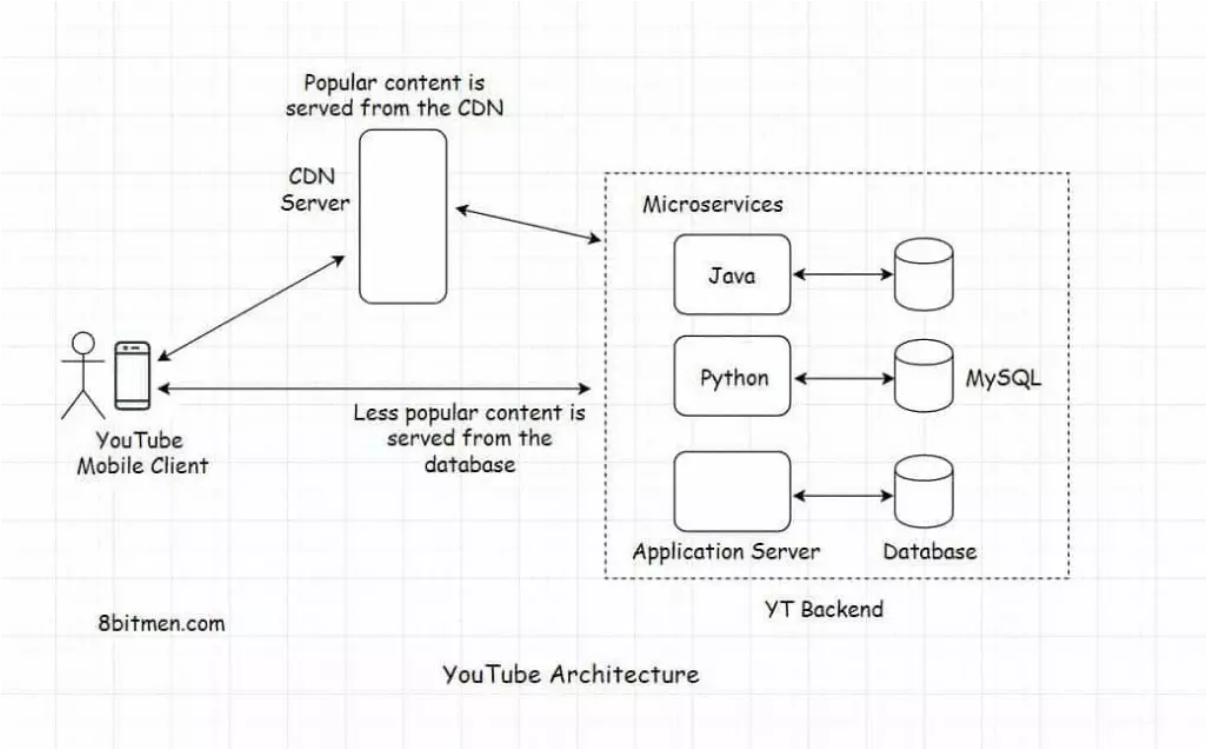
2. L'évolution de YouTube avec l'architecture micro-services

Si YouTube a envisagé une architecture microservices, c'est aussi en apprenant de ses concurrents, en particulier de la plateforme Netflix spécialisée dans des services de streaming (vidéo à la demande) de films et de séries [13]. Les deux entreprises proposent des services similaires et font face à des enjeux communs, tels que le stockage de contenu vidéo en grande quantité et de taille importante, la lecture de vidéos et l'utilisation intensive par un très grand nombre d'utilisateurs. Netflix emploie une architecture microservices depuis 2012 après avoir été confronté à des problèmes majeurs liés à une corruption de sa base de données en 2008 [13]. En effet, la société employait une architecture monolithique avant cela, c'est-à-dire une architecture comme celle en couches où les composants sont interdépendants. L'entreprise se trouvait donc en difficulté lors de la résolution du problème, étant donné que la base de données était liée à plusieurs services, ce qui nécessitait d'analyser le système en entier pour identifier et réparer le problème sans impacter l'expérience des utilisateurs et la synchronisation entre les couches fonctionnelles [6]. Afin de répondre à cela, Netflix a engagé un long processus de migration de ses services durant plusieurs années. YouTube avait donc une source d'inspiration pour changer d'architecture.

D'autre part, la plateforme YouTube ayant beaucoup évolué depuis sa création, a procédé au fil du temps à l'implémentation de nouveaux services basés sur des algorithmes avancés (recommandation personnalisée, détection de copyrights et modération de contenu par machine learning, etc.). Ces derniers ont été développés en tant que microservices pour plus de simplicité : l'algorithme de recommandation par exemple peut gérer avec sa base de

données (grâce au partitionnement de la base de données principale) dédiée aux préférences de visionnage des utilisateurs, qui lui concerne et sont réellement utiles à lui, ce qui améliore considérablement les performances par rapport à l'utilisation d'une base de données unique et partagée entre tous les services. Ces deux plateformes utilisent désormais des microservices pour divers services tels que l'encodage et le transcodage, ainsi que la recherche et la lecture de vidéos [13] [14].

En termes de scalabilité, l'architecture microservices permet donc à YouTube d'implémenter de nouvelles fonctionnalités en plus de celles déjà existantes, sans avoir la nécessité de modifier les services de base, à condition qu'elles soient configurés correctement pour une intercommunication via des APIs. Leur indépendance favorise l'évolution des services, leur modification et leur maintenance. Cependant, il est pour autant possible que YouTube fait usage d'une architecture hybride qui préserve certains éléments de celle en couches, tout en favorisant les microservices.



Source illustration : Article "YouTube database – How does it store so many videos without running out of storage space?" sur le blog scaleyourapp.com

Conclusion

La plateforme YouTube est une application qui a débuté avec des services basiques de diffusion de vidéos en ligne, en suivant une idée de simplicité avec l'usage d'une architecture en couches. Au fil du temps, un très grand nombre de changements ont été effectués à travers l'implémentation de nouveaux services et fonctionnalités qui ont complexifié la structure et donc la gestion de l'application, ce qui a nécessité une certaine restructuration de l'application et du processus de développement afin de permettre son évolution à travers notamment une architecture micro-services.

Bibliographie

[1] - Article Wikipédia sur l'entreprise YouTube - https://fr.wikipedia.org/wiki/YouTube

[2] - Statistiques 2023 Youtube (27 février 2023), Article GlobalMediaInsight - https://www.globalmediainsight.com/blog/youtube-users-statistics/#:~:text=YouTube%20every%20day.-,Monthly%20Active%20Users%20on%20YouTube,2.6%20billion%20monthly%20active%20users

[3] - Diverses réponses communautaires à la question "What is the software architecture of YouTube?" sur quora.com - https://www.quora.com/What-is-the-software-architecture-of-YouTube/answer/Turkishcouk-1 https://www.quora.com/What-is-YouTubes-architecture/answer/Jake-Cook-8 https://www.quora.com/What-is-the-software-architecture-of-YouTube

[4] - Questions et réponses en utilisant ChatGPT

[5] - Article "YouTube database – How does it store so many videos without running out of storage space?" sur le blog scaleyourapp.com - https://scaleyourapp.com/youtube-database-how-does-it-store-so-many-videos-without-running-out-of-storage-space/

[6] - Cours de notre enseignant M. Osmani sur les architectures logicielles, particulièrement sur celle en couches - https://lipn.univ-paris13.fr/~osmani/sa/coursAL-2.pdf

[7] - Cours openclassrooms sur l'architecture en couches - https://openclassrooms.com/fr/courses/7210131-definissez-votre-architecture-logicielle-grace-aux-standards-reconnus/7371476-apprenez-larchitecture-en-couches

[8] - Article "Introduction to the Layered Architecture (n-Tier Architecture)" sur brcline.com - https://www.brcline.com/blog/introduction-to-the-layered-architecture-n-tier-architecture#:~:text=Companies%20like%20Salesforce%2C%20NetSuite%2C%20Oracle,of%20persistence%20or%20Database%20Layer.

[9] - Article "7 Years Of YouTube Scalability Lessons In 30 Minutes" sur highscability.com - http://highscalability.com/blog/2012/3/26/7-years-of-youtube-scalability-lessons-in-30-minutes.html

[10] - Article "System Design YouTube" sur enjoyalgorithms.com - https://www.enjoyalgorithms.com/blog/design-youtube-system-design-interview-question

[11] - Article sur l'histoire du framework Vitess - https://vitess.io/docs/15.0/overview/history/

[12] - Documentation Azure sur le "Style d'architecture orientée microservices" - https://learn.microsoft.com/fr-fr/azure/architecture/guide/architecture-styles/microservices

[12] - Article Wikipédia sur l'entreprise Netflix - https://fr.wikipedia.org/wiki/Netflix

[13] - Article "Understanding Microservices Architecture at Netflix" sur sayonetech.com - https://www.sayonetech.com/blog/microservices-netflix/

[14] - Article "Les microservices, qu'est-ce que c'est ?" sur redhat.com - https://www.redhat.com/fr/topics/microservices/what-are-microservices