Spotify : le catalogue musical multiplateforme

Sommaire

I. Présentation de Spotify

Spotify, le service suédois de streaming musical, a été fondé en 2006 par Daniel Ek et Martin Lorentzon, révolutionnant l'industrie musicale en proposant un accès à des millions de morceaux du monde entier. Sa plateforme, lancée publiquement en 2008, a rapidement gagné en popularité en offrant une alternative à l'achat de musique individuelle sur des plateformes comme Amazon et iTunes. Avec son modèle gratuit soutenu par des annonces et une version Premium sans publicité, Spotify a établi le streaming comme une norme pour les amateurs de musique, dépassant ses concurrents tels qu'Apple Music et YouTube Music.

Aujourd'hui, Spotify compte 345 millions d'utilisateurs actifs, dont 155 millions d'abonnés Premium. Les utilisateurs bénéficient de fonctionnalités telles que la personnalisation des profils, la création de playlists et la possibilité de partager leurs découvertes musicales avec leurs amis. Spotify propose des playlists adaptées aux goûts de l'utilisateur, créant ainsi une expérience musicale personnalisée.

II. Architecture logicielle de Spotify

a. Présentation de l'architecture logicielle de Spotify

L'application Spotify repose sur une architecture logicielle Microservices. Chaque Microservice n'a qu'une seule responsabilité simple et dans la plupart des cas ils disposent d'une base de données privée avec sa propre logique sur laquelle aucun autre processus ne peut intervenir. Ce qui permet de diviser les aspects de l'application en simplifiant ainsi la maintenance, la gestion, l'évolution et l'extension du logiciel. De plus, cette architecture favorise une collaboration accrue entre les équipes de développement en leur permettant de se concentrer sur des aspects spécifiques de l'application sans perturber les autres et en réduisant la durée du développement.

Cela donne à l'application la possibilité d'évoluer rapidement en déployant différents services sur différentes machines avec des performances personnalisées en fonction de la demande de chaque service.

Spotify se base sur les trois entités suivantes :

Composants : Il s'agit de pièces individuelles appelées composants du logiciel telles que le service backend, le site web ou le pipeline de données. Cette couche est également appelée couche d'interface utilisateur, elle est responsable de l'interaction avec l'utilisateur. Exemple :

- le composant dédié à la lecture de musique : il est responsable de la gestion du streaming audio pour les utilisateurs. Ce microservice gère la lecture des morceaux, la mise en file d'attente, les contrôles de lecture (lecture, pause, saut de piste, etc.) et la gestion des playlists en temps réel.
- La recommandation personnalisée. Ce microservice utilise des algorithmes d'apprentissage automatique pour analyser les habitudes d'écoute des utilisateurs, leurs préférences musicales et d'autres données pour recommander des morceaux, des artistes et des playlists adaptés à chaque utilisateur.

APIs : Cette couche de traitement fait le lien entre les différents composants du logiciel et définit l'interface entre ces composants. Elle gère les fonctionnalités métier de l'application et traite les données des différents composants du logiciel.

Ressources : Cette couche représente l'infrastructure nécessaire pour faire fonctionner un composant en temps d'exécution, comme les bases de données, les machines virtuelles ou les systèmes de fichiers. Elle est responsable de la gestion des données de l'application, en stockant et en récupérant les données de la base de données.

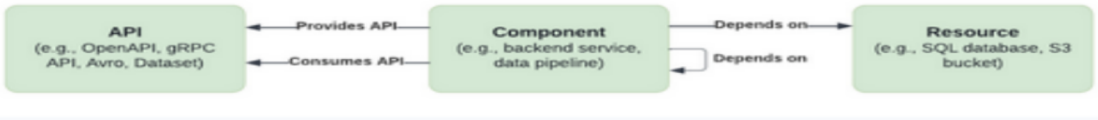

Figure 1. Relationship between core entities.

b. Choix de l'architecture logicielle de Spotify

Spotify utilise une architecture logicielle Microservices pour plusieurs raisons importantes:

Séparer les préoccupations : L'architecture en Microservices permet de séparer clairement les différentes responsabilités et fonctionnalités de l'application. Chaque couche a un rôle distinct et des responsabilités définies, ce qui facilite la compréhension du système dans son ensemble et simplifie la maintenance.

Faciliter la maintenance du logiciel : En divisant l'application en différentes couches, il devient plus facile de gérer et de maintenir chaque composant séparément. Cela permet aux équipes de développement de se concentrer sur des parties spécifiques de l'application sans perturber les autres, ce qui améliore l'efficacité et la productivité du développement logiciel.

Gérer l'évolutivité et l'extensibilité du logiciel : L'architecture microservices rend l'application plus évolutive et extensible. Les nouvelles fonctionnalités peuvent être ajoutées ou modifiées plus facilement sans affecter les autres parties de l'application. Cela permet à Spotify de s'adapter rapidement aux besoins changeants des utilisateurs et du marché.

III. Évolution de Spotify

Auparavant, jusqu'en 2014, le système de distribution de la musique reposait sur un modèle hybride pair-à-pair et client-serveur, permettant de réduire la charge des serveurs centraux.

Au fil des années, Spotify a évolué techniquement en adoptant une architecture microservices. Cette transition lui a permis de mieux gérer la complexité croissante de sa plateforme et de faciliter le déploiement indépendant de nouvelles fonctionnalités. Voici quelques exemples de microservices que Spotify utilise actuellement.

- Un microservice dédié à la lecture de musique: il est responsable de la gestion du streaming audio pour les utilisateurs. Ce microservice gère la lecture des morceaux, la mise en file d'attente, les contrôles de lecture (lecture, pause, saut de piste, etc.) et la gestion des playlists en temps réel.

- La recommandation personnalisée. Ce microservice utilise des algorithmes d'apprentissage automatique pour analyser les habitudes d'écoute des utilisateurs, leurs préférences musicales et d'autres données pour recommander des morceaux, des artistes et des playlists adaptés à chaque utilisateur.

Concernant la représentation de l'architecture logicielle de Spotify, elle a été pendant longtemps constituée de simples boîtes et flèches disposées sur un ou plusieurs tableaux blancs. Cependant, cette façon de modéliser était peu conventionnelle et représentait certains défis.
Afin de rendre la modélisation du fonctionnement du logiciel Spotify compréhensible, Spotify s'est inspiré du modèle C4 pour créer son propre modèle. Il s'agit d'un modèle de visualisation de l'architecture logicielle d'un système reconnu de nos jours, et conventionné.
Voici les différents **points de vue** du modèle de Spotify :

- Le diagramme de **contexte,** qui apparaît dans le modèle C4. Il représente les interactions du système global avec des systèmes extérieurs ou des utilisateurs.
- Le diagramme des composants, qui apparaît dans le modèle C4 sous le nom de diagramme de **conteneurs**. Ce diagramme représente les différents conteneurs d'un système, et leurs relations. Un conteneur est un sous-système qui peut fonctionner de façon indépendante.
- Le diagramme de **paysage**, qui est spécifique au modèle de Spotify. Le diagramme de paysage du système décrit l'ensemble des relations entre les systèmes, la façon dont ils sont reliés et leurs dépendances avec des systèmes externes.

IV. État de l'art de spotify

1. Technique : Architecture logicielle des concurrents

De nos jours, Spotify reste le leader du marché de streaming musical dans le monde.
Parmi ses principaux concurrents, nous pouvons citer Apple Music, Deezer, Youtube Music, Qobuz, Tidal ou encore Amazon Music.

Deezer a une architecture orientée événements basée sur Kafka. L'architecture orientée événements utilise trois éléments principaux, à savoir le créateur d'événement, le routeur d'événement et le consommateur d'événement.
Apple Music repose sur une architecture trois tiers constituée d'une couche base de données, d'une couche de présentation du logiciel, et d'une couche de traitement pour mettre en oeuvre les règles de gestion.

2. Évolution technologique du streaming musical

L'évolution des plateformes de streaming musical est très axée sur la personnalisation de l'expérience l'utilisateur, en lui créant un profil adapté, basé sur ses données collectées. Ainsi, la plupart des plateformes ont recours à des algorithmes de recommandations de plus en plus performants. Nous pouvons citer par exemple, la création de playlists mélangeant celles de plusieurs utilisateurs (Jam de Spotify ou Shaker de Deezer), la génération par intelligence artificielle d'une playlist adaptée à nos goûts, ou encore le jeu Blind Tests de Deezer.
Des langages comme Java, Python et JavaScript sont couramment utilisés pour le développement des applications et des services de streaming musical.

V. État de l'art business

Les services de streaming musical reposent sur le business model Freemium. Cela signifie que ces services proposent une version gratuite de leur service, souvent accompagnée de la diffusion de publicités, afin d'attirer un grand nombre d'utilisateurs, puis de convertir ces utilisateurs en clients qui vont adhérer à la version payante du service, sans publicité. L'essai gratuit est l'étape qui permet aux utilisateurs de se familiariser avec le service, et d'être attirés par la version payante. Cet essai gratuit génère du revenu à ces services de par la diffusion de publicités.
Parmi les concurrents cités, seuls Apple Music et Qobuz ne proposent pas de version gratuite, et ne diffusent pas de publicités. Le prix mensuel de l'abonnement classique à ces services est généralement de 10,99 € (14,99 € pour Qobuz). Néanmoins, il existe des formules d'abonnement pour les étudiants, pour les duos et pour les familles, chez Spotify, Amazon Music, Deezer, etc.

Conclusion

En conclusion, nous pensons que l'architecture la plus adaptée pour le streaming musical est celle des microservices. Comme le démontre le cas de Spotify, cette architecture permet une gestion efficace de la complexité croissante de la plateforme et facilite le déploiement indépendant de nouvelles fonctionnalités. Ces avantages sont cruciaux pour

offrir une expérience utilisateur fluide et innovante. C'est cette adoption des microservices qui a contribué à faire de Spotify la première application dans le domaine du streaming musical, établissant un standard de performance et de fiabilité difficile à égaler.

Sources

Architecture iOS : https://www.quora.com/What-is-iTunes-architecture
Wikipédia de Spotify : https://fr.wikipedia.org/wiki/Spotify#Fonctionnement_technique
Diagramme de conteneurs : https://sellugsk.live/product_details/57981653.html
Site de Spotify pour les développeurs : https://engineering.atspotify.com/2022/07/software-visualization-challenge-accepted/
Tarifs Spotify : https://www.spotify.com/fr/premium/#plans
Tarifs Deezer : https://www.deezer.com/fr/offers/?gad_source=1&gclid=EAIaIQobChMIlvXR_8XihQMV6ToGAB056gEvEAAYASAAEgKq-PD_BwE
Amazon Music Unlimited offres : https://www.amazon.fr/music/unlimited/?_encoding=UTF8&ref_=sv_dmusic_1
Comparaison avec les concurrents : https://www.frandroid.com/android/288776_streaming-musical-offre-choisir
Wikipedia de Freemium : https://fr.wikipedia.org/wiki/Freemium
Business Model de Spotify : https://pswd.fr/modele-economique-spotify/
Services de streaming musical gratuit : https://www.monassistantnumerique.com/article/top-10-des-sites-applications-pour-ecouter-de-la-musique-gratuitement
Deezer a une architecture orientée événements (expliqué par Vincent Lepot, Lead Web Architect et Backend SRE Enabler chez Deezer) : https://fr.slideshare.net/VincentLepot/event-driven-architecture-comment-deezer-passe-en-mode-ractif
Linkedin de Vincent Lepot : https://www.linkedin.com/in/vincent-lepot-3013625/
Wikipedia du Cloud Computing : https://fr.wikipedia.org/wiki/Cloud_computing
Amazon Music : https://aws.amazon.com/fr/solutions/case-studies/amazonmusic-amplify-appsync-case-study/
Architecture Orientée événement selon IBM : https://www.ibm.com/fr-fr/topics/event-driven-architecture
Algorithme de recommandations Spotify : https://digital.hec.ca/blog/algorithme-spotify/

Commande vocale Spotify :
https://www.presse-citron.net/spotify-developpe-une-technologie-capable-de-definir-votre-humeur-selon-votre-voix-pour-vous-proposer-des-playlists-adequates/
Jam sur Spotify :
https://www.tech-generation.fr/2023/09/26/spotify-apporte-t-il-une-veritable-innovation-avec-jam/
Présentation de Spotify d'un groupe de BUT2 informatique de l'année précédente :
https://docs.google.com/document/d/1pmjFQ5BJXDHI0JNdt1UPGxJhYa-m1rt-lae2vjfEHVI/edit#