Migrer ses APIs vers GraphQL : pourquoi ? comment !

Speakers : Guillaume Scheibel (Expedia)
Format : Conférence
Date : 19 avril 2019

Guillaume présente le travail réalisé chez Expedia depuis 9 mois : passer leurs APIs en GraphQL.
Le site de réservation de voyage Expedia existe depuis 25 ans. Guillaume va se concentrer sur les API appelés par les utilisateurs.
Expedia est parti de Microsoft C++, est passé par Java et utilise d'autres stacks techniques.

Quelques chiffres en 2019:

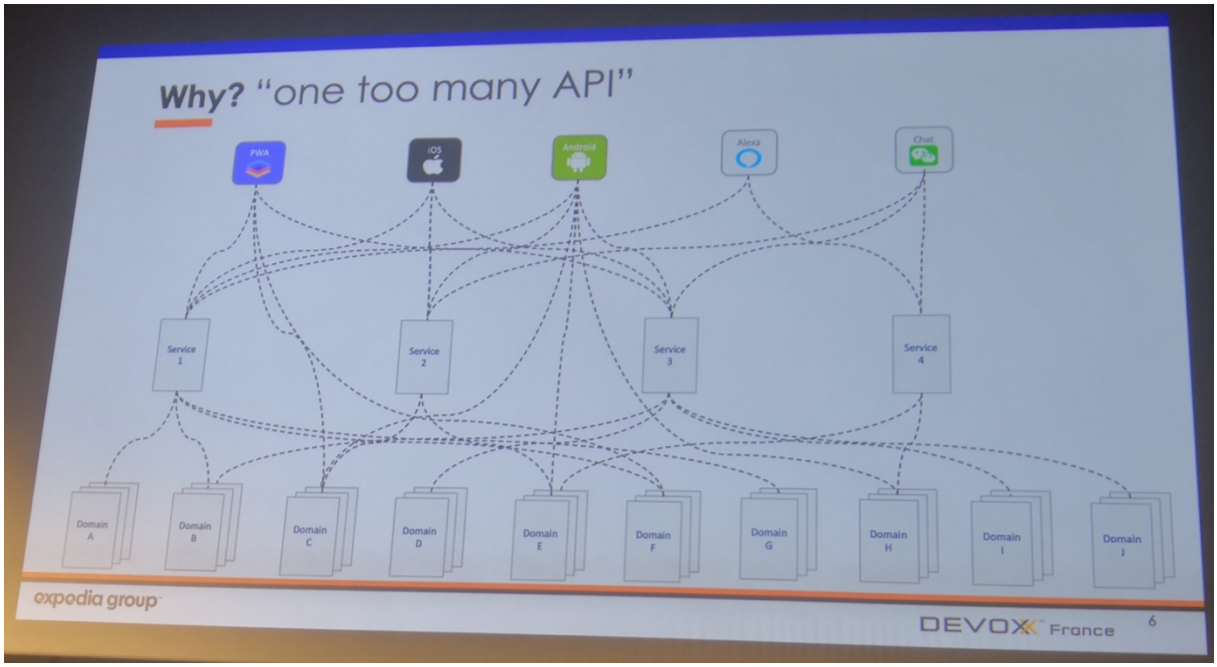
- 500 développeurs à travers le monde
- 20 API pour les hôtels, les vols, les réservations. Expedia sort du monolithe.
- 50 applications front-end (iOS, Android, PWAs) : l'application de recherche de vol n'est pas la même que celle de recherche d'hôtel

Problèmes rencontrés :

- API illisible, pas documentée
- Chaque device utilise des API différentes : un changement doit être appliqué sur toutes les API. De ce fait, implémenter la RGPD a été un énorme chantier.
- Inconsistance et incohérence : les formats de date ou de devise changent d'une API à l'autre.

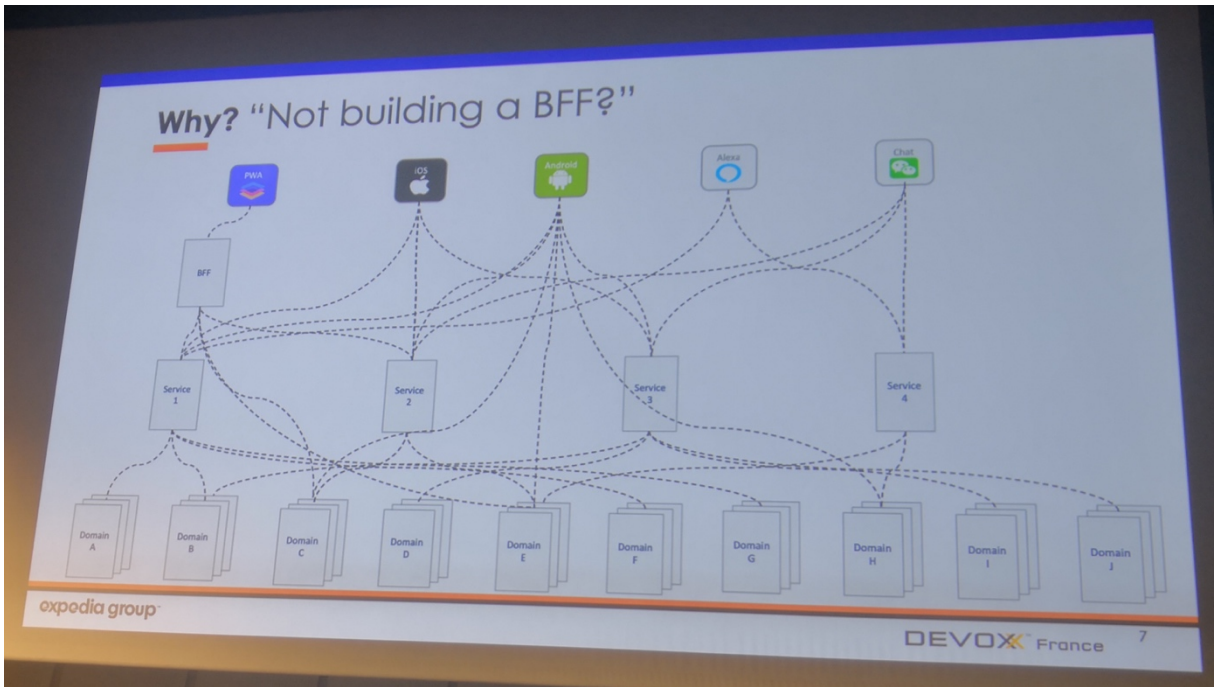
Les utilisateurs sont mécontents

Architecture très simplifiée d'Expedia :



La loi de Conway : le design des systèmes reflète l'organisation. Monstruosité : une équipe = une stack. Les équipes ont besoin d'autonomie, mais elles ne communiquent pas.

Pattern Backend For Frontend (BFF) démocratisé par Spotify : une seule API à intégrer en tant que front, plus facile. Le problème qui se pose : faut-il faire un BFF pour toutes les applis front-end ? Non, car cela entrainerait des points de contention, le code deviendrait un monolithe, et même un monolithe distribué dans lequel toutes les applis doivent être releasées en même temps.
Le BFF réduit le nombre de endpoints à intégrer, mais ne résout pas le problème de cohérence et de cohésion.



Chez Expedia, problème récurrent sur les dates : quel format adopter ?

- yyyy-MM-dd
- MM/dd/yyyy
- dd/MM/yyy
- ISO-8601

Parfois, 2 champs de la même réponse n'ont pas le même format.

Même problématique sur les prix : €123.45, 123.45EUR, <EUR, 123,45>

How ?

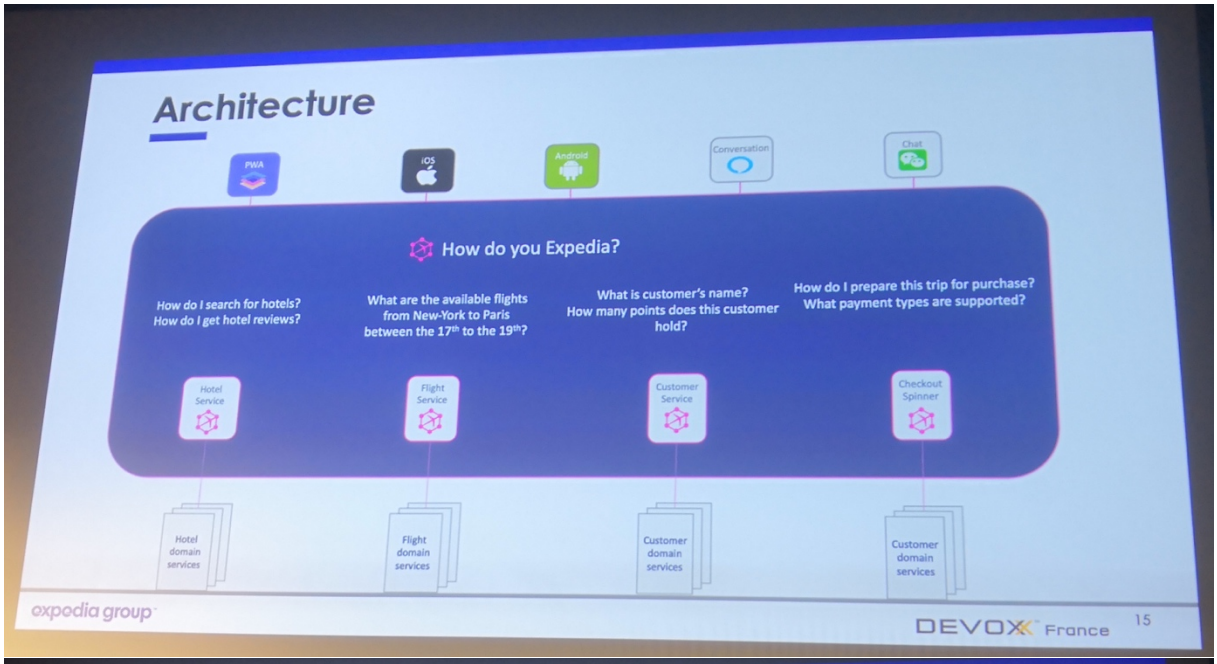
Expédia cherche à créer une **cohesive API** (cohérente)

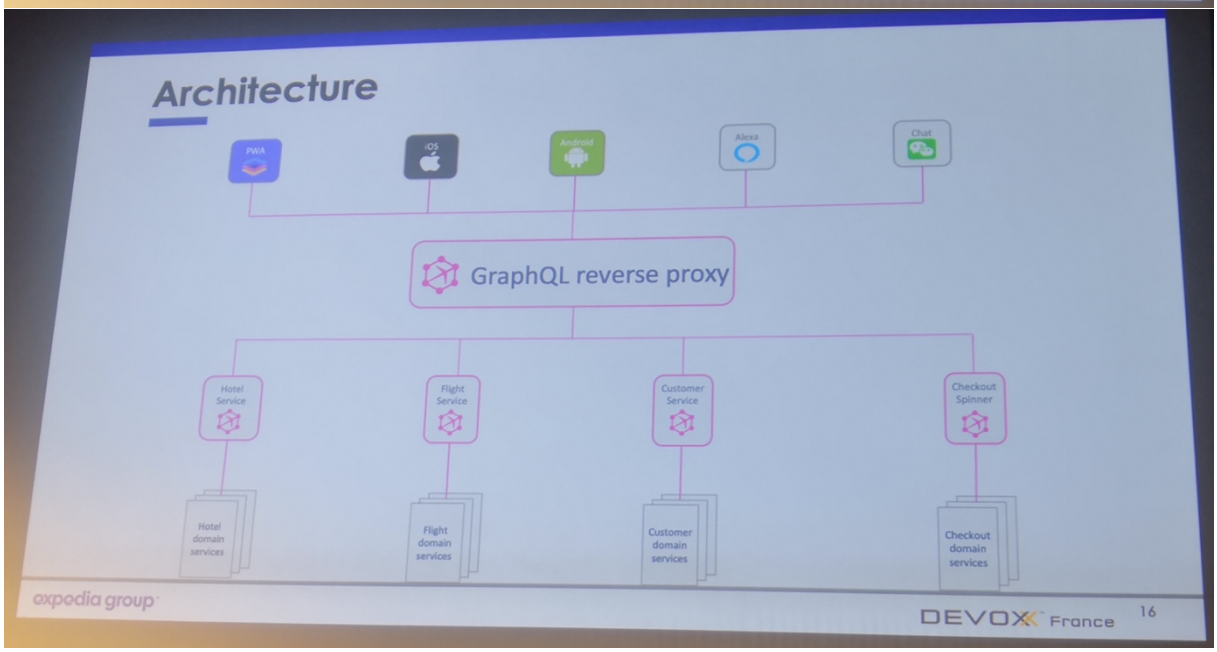
- Partager la même technologie
- L'API devient un produit et n'est plus seulement un élément technique
- Flexible pour servir des clients hétérogènes (mobile, Alexa …)

Comme contrat d'interface, Expédia utilise désormais des **schémas**. L'API doit répondre aux questions des clients. Les développeurs backend ne doivent pas exposer directement leur domaine. Le domaine d'Expedia est très complexe.

GraphQL

GraphQL a été créé par Facebook en 2012 puis a été cédé à la [fondation GraphQL](). GraphQL propose un schéma.
Principe du WYSIWYG : on sélectionne les champs que l'on souhaite. Par exemple, Alexa n'a pas besoin des images pour sa synthèse vocale.





Chaque domaine expose désormais une API GraphQL. On retrouve un service par domaine.
Le reverse proxy GraphQL ne contient aucune logique métier.
Le reverse proxy concatène tous les schémas des services sous-jacents. Au final, les applications front ne voient qu'un seul schéma.

Le proxy est déployé dans Node.JS et repose sur la solution Apollo server.
Côté back, ils ont créé GraphQL-Kotlin :

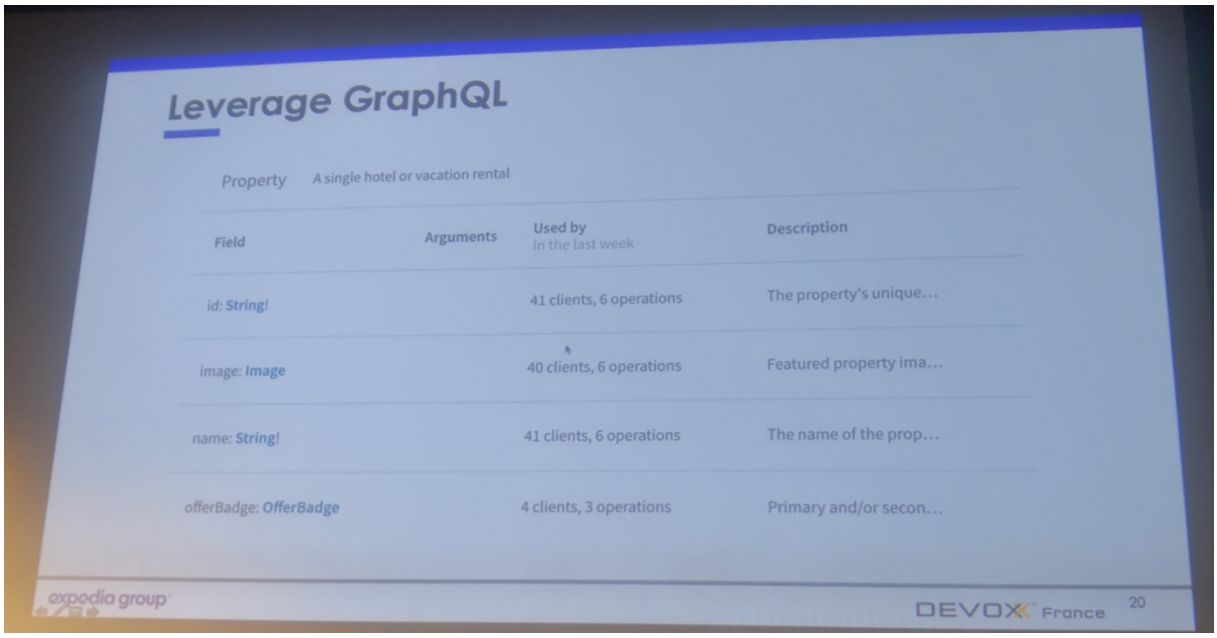
- Open-source
- Code first : génération du schéma à partir des types Kotlin
- Délègue l'exécution à GraphQL-java

S'ensuit une démo basée sur Spring Webflux et les coroutines

En GraphQL, il existe 3 types d'opérations : requête, mutation et streaming.
Kotlin se marie très bien avec GraphQL car les 2 permettent d'exprimer qu'un champ peut-être null ou pas.

Utilisation de de l'IDE Playground : swagger en mieux d'après le speaker.
Playground utilise le schéma GraphQL pour l'auto-complétion.
Le schéma GraphQL permet de déprécier un champ (annotation @Deprecated dans Kotlin).
Possibilité de rajouter de la documentation avec @GraphQLDescription

GraphQL propose de l'instrumentation : par exemple, il permet de tracer l'usage des champs dépréciés, pour savoir si des clients l'utilisent encore. GraphQL propose également tout un tas de métriques pour savoir qui appelle quoi, au niveau du champs :



Guillaume évoque la possibilité d'utiliser les statistiques de Production pour valider qu'une Pull Request ne casse pas le schéma utilisé par un client.