Votre API passe-t-elle les 50 points du contrôle technique ?

Speakers : François-Guillaume Ribreau (Ouest-France)
Format : Conférence
Date : 19 avril 2019

Historique des API

François Guillaume commence son talk par une citation de l'information Christopher Strachey (1969) qui a travaillé avec Turing : « Je pense que les sciences informatiques vont devenir importante. Ne pas transmettre l'état de l'art, mais plutôt se concentrer sur les principes. »

L'histoire des API remonte aux débuts de l'informatique :

- 1960' : API d'une librairie
- 80-90' : accès distant à des méthodes
- 2000' : REST par Roy Fielding
- 2010' : API publiques, monétisées

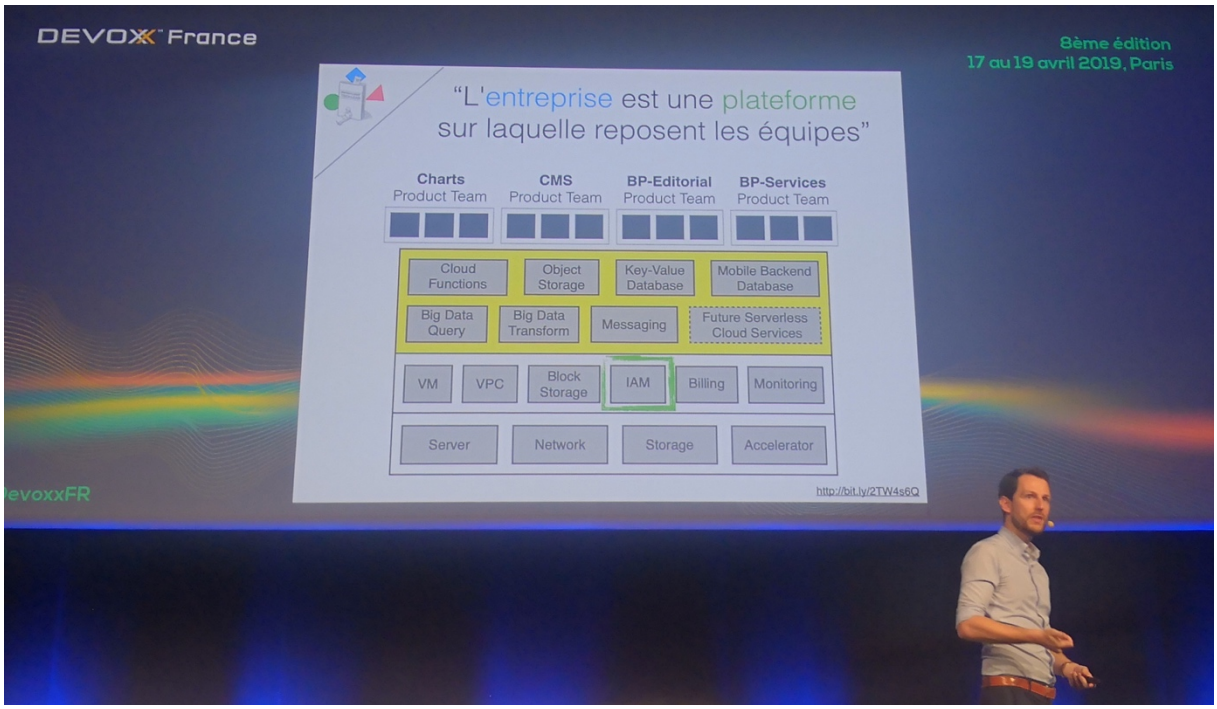
Qu'est-ce qu'un contrôle technique dans le monde auto ? 131 points de contrôle et 410 défaillances.
Dans l'ingénierie, on établit des checklists pour compenser les limites de la mémoire.
Sur une API (au sens large et pas que web), il y'a plus de 70 points de contrôle.
Souhaite-t-on prendre en compte toutes ces contraintes lors du démarrage du développement d'une nouvelle API ? François-Guillaume est convaincu qu'il faut en prendre en maximum. A force de réaliser des API, on agrandit son état de connaissance et on peut donc se rajouter des contraintes.

Avant les développements, il faut se demander quelles sont les objectifs d'une API ?
Service Level Objectives (SLO)
Les objectifs doivent être mesurables, avoir des limites hautes et basses.
Exemple : « la latence de 95% des requête soit inférieure à 50ms »

Service Level Agreements (SLA)
Contrat implicite ou explicite. Slack fait une ristourne le mois suivant si SLA non respecté.

Définir le nom du service
Ne serait-ce que pour créer le nom du repository. Nommer les choses est difficile en informatique. Choisir un nom fonctionnel.
« Le nommage doit être ennuyeux » : descriptif, transparent, non ambigu. Vient du clean code.

Utiliser un Identity & Access Management service (IAM)



L'IAM a 3 responsabilités : l'authentification, l'autorisation et la traçabilité
Comme le montre la photo ci-dessus, l'IAM est très bas dans la couche du SI.
Keycloak est une solution Open Source Java supportée par RedHat.
A chaque requête, on envoie le token à l'IAM qui vérifie si la requête à le droit de faire tel appel. L'appel est stocké dans l'audit.
Modèle de sécurité par périmètre des années 1990 (château fort).
Dans les années 2000, il y'a du SaaS, des applis mobiles. L'usage du VPN se démocratise.
Avec un VPN, on rentre dans le château fort par le pont levis.
Il devient désormais nécessaire de considérer que le réseau n'est plus fiable.
Beaucoup de SI sont encore en untrusted network.
En 2019, que ce soit des middleware IAM ou Istio, on peut le mettre en place simplement.

Versionning
Gérer le versionning dès le début avec le semantic versionning.
Le client doit pouvoir choisir la version de l'API.
Exemple avec HTTP : GitHub utilise le header Accept
Il existe d'autres approches : version dans l'URL
Autre approche de Stripe.com : La version en prod est toujours la dernière version. Outil de détection puis de down-migration pour passer de la V3 à la V2.

Définir et s'assurer de la compatibilité descendante
L'ajout d'un champ peut casser le parseur du client.
Projet github.com twitter/diffy : proxy permettant de comparer les résultats d'appel d'une même requête à l'égard de plusieurs versions.

Support métier

Pagination, search, sorting, filtering, field selection, field expansion.
Pagination : souvent on choisit d'implémenter la pagination plus tard. Le risque est qu'en production l'API renvoie toute la base et fasse tomber le système. Nécessiter de « tout limiter dans l'espace et dans le temps ». La pagination limite tout dans l'espace (ex : max 1000 éléments).

Fingerprints
Attention à ne pas renvoyer les fingerprints
Exemple : numéro de version de Nginx datant d'il y'a 2 ans et donc propice aux attaques

Split state & logic
Tout comme les langages de programmation, on passe d'une logique de mutabilité à l'immutabilité. Les données peuvent être stockées de manière immutable. Permet d'historiser les données et de faire des retours arrière.

Support du multi-tenant
La Production est le seul environnement public d'une équipe. L'environnement de dév ou de staging sont privés.
Principe : « les équipes pointent sur la production des autres ».
Notion d'App dans Twiter, de royaumes dans IAM.
Chez Ouest-France, ils ont plusieurs sites de prod.

Utiliser une signature
Ajout d'un hash ajouté à l'URL et qui sert de signature. Ajout également du clientId ou tenantId.

Mettre en place des timeouts
Lorsqu'on appelle une base de données, ajouter systématiquement un timeout. Rend explicite la possibilité d'une défaillance réseau. On doit alors ajouter du retry ou un circuit breaker.

Tests
Écrire des tests de post déploiement.
Utilisation de smoke-tests : tests fonctionnels utilisés à la fois sur staging et en prod. Permet de détecter les erreurs avant les utilisateurs finaux.

Limiter le nombre maximum de requêtes
Définir des quotas
Amazon et Google définissent systématiquement des quotas, ceci afin de monétiser leurs API.

Mettre en place des alertes
Lorsqu'on crée une métrique, il est intéressant de définir systématiquement une alerte. Au

début, on est spammé. Mais on peut affiner ses seuils.
Les run books : étapes à mettre en place pour résoudre le problème ou comment investiguer
GitLab met en open source la manière dont ils gèrent leurs run books.

Documenter son API
Approche de générer la doc à partir des tests

Déployer son API
Dark-launch : on prend son trafic de production et on l'envoie sur staging pour étudier le comportement.