Architecture Logicielle: Construction

Equipe B: Guillaume Piccina, William d'Andrea, Nicolas Fernandez, Yann Brault

Sujet V6 : Pay as you pollute

Ce document traite de l'architecture et de son évolution chaque semaine.

Semaine 42: 18_10

Les modifications de l'architecture:

- Ajout d'un bus événementiel entre le user et les services de tracking
- Modification de l'interface entre le système de caméra et le système, qui passe désormais par un service dédié plutôt que par Car tracker
- Billing perd la responsabilité de récupérer en BD les infos d'un user qui s'arrête et c'est désormais tracking shutdown qui le fait et transmet les infos à billing

Détails des modifications :

- Le bus vient se placer pour fonctionner de la manière suivante: un user envoie émet un événement dans le bus lorsqu'il se déplace. Les services de tracking (car tracker et tracking shutdown) consomment le message en fonction de qui est concerné par cet événement et exécutent ensuite leur logique propre.
- Le bus va également être utilisé à des fins de configuration. C'est à dire que si le car tracker remarque qu'un user est proche d'une limite entre 2 zones il lui enverra un événement de configuration pour que l'utilisateur envoie sa position plus fréquemment.
- Un service position checker a été ajouté pour faire l'interface entre le système et le service externe de caméra. Ce service est connecté au bus également et peut consommer un événement de déplacement (notamment dans le cas où l'utilisateur démarre à nouveau après un arrêt) afin de vérifier que la position d'arrêt et de démarrage sont les mêmes.
- Dorénavant c'est le service tracking shutdown qui, une fois qu'il reçoit un notification d'arrêt d'un utilisateur, va aller récupérer en BD les infos de déplacement de l'utilisateur et les transmettre à billing pour la rédaction de la facture.

Les services en détails:

- **Car tracker :**
    - Consomme dans le bus un topic de tracking
    - Transmettre à grande vitesse les infos de tracking vers la BD (heure de pointes, bouchons)
    - Doit attendre le retour de zone pollution avant d'envoyer les data dans la base (y a un lock ici)
- **tracking shutdown :**
    - Consomme dans le bus un topic de shutdown
    - Récupère en db *Tracking_infos* les data du user
    - Déclenche le billing pour un user en envoyant les data
    - flush la db de ces data
- **Position checker**
    - Consomme dans le bus un topic de redémarrage
    - Communique avec le système externe de caméra pour surveillance
    - contact billing en cas de problème
- **Zones pollution :**
    - Doit tirer à intervalle régulier les infos de pollution et les mettre en cache (solution de stockage clé valeur avec très gros débit de lecture)
    - Doit trouver une correspondance entre la position gps et la zone pour renvoyer la zone à car tracker
- **Billing :**
    - Reçoit les infos de tracking depuis tracking shutdown
    - Calcul la facture finale
    - Fait l'interface avec la banque pour le paiement.

Scénario du flot de données:

- L'utilisateur envoie son id, sa position gps et un timestamp dans le bus, il reçoit sa zone de circulation en réponse à sa requête ce qui lui permet de calculer une estimation du coût de son trajet. Il peut aussi recevoir au travers du bus un message de configuration pour lui dire d'envoyer sa position plus fréquemment
- Le service de tracking consomme les infos dispo dans le topic de tracking du bus et envoie la position gps au service de pollution qui lui renvoie une zone de pollution, puis le tracker envoie les 4 informations au stockage.
- Quand l'utilisateur finit son trajet, il envoie les mêmes informations que d'habitude sauf que cette fois-ci le service tracking shutdown sera le consommateur.
- Le service tracking shutdown, en consommant les infos de son topic, reçoit l'info que l'utilisateur s'est arrêté, alors il va chercher en DB les infos de tracking de l'utilisateur et les envoie vers le billing
- Le service de billing avec les infos qu'il a reçu va calculer la facture puis il va envoyer la facture au service de banque
- L'utilisateur recevra sa facture directement de sa banque

Choix de techno :

- Bus : RabbitMQ pour le moment (évolution possible en AL Evolution)
- BD principale : NoSQL
- BD zone pollution: SQL qui supporte une grosse charge de lecture
- Archi micro service : NestJS

Enfin voilà un schéma de l'architecture: