Rapport 1 : Dronazone

Alexandre Bolot - Hugo François - Grégoire Peltier - Prune Pillone

Interprétation du sujet

Quelques hypothèses ont été émises sur le fonctionnement du projet et donc sur ses fonctionnalités et limitations :

- Il n'y a qu'un seul entrepôt dont la localisation est connue.
- Le stock de l'entrepôt est illimité
- Un drone prend une commande dès que possible (FIFO)
- Un drone ne transporte qu'un paquet
- Pas de contrainte de poids ni de taille des paquets comparés au drone.
- Le passage de flotte d'un drone (active, en charge etc), ne se fait pas automatiquement mais à l'action de Elena
- Le drone retient en interne sa position d'origine, où il doit retourner, par un mécanisme interne à son démarrage
- Les produits sont stockés dans le même entrepôt d'où partent les drones.

Au niveau technique, nous avons choisi d'utiliser :

- Java Spring. Largement utilisé et correspondant au besoin d'accès aux données et de création de service, ceci nous paraissait le choix idéal.
- Postgres. Référence dans les bases de données et respecte les normes.
- Cucumber pour la validation des User Story.

Explication de l'architecture

Nous avons conçu notre architecture en prenant en compte plusieurs axes qui selon nous définissent les enjeux de la solution nécessaire à Dronazone :

- Une bonne adaptabilité aux fluctuations de charge utilisateur, que ce soit du point de vu du nombre clients (représentés par Roger), du nombre de commandes ou du nombre de techniciens gérant ces commandes (représentés par Klaus).
- Nous devons également avoir une bonne répartition des responsabilités de notre système étant donné qu'il intègre plusieurs domaines "métier" : commerce en ligne, gestion de colis et gestion d'une flotte de drones. Chacun de ces domaines a des besoins qui lui sont propres et il faut pouvoir adapter chaque partie de notre système à ces besoin, d'un point de vue fonctionnel et technique.

Pour cela nous proposons une architecture composée de trois services distincts, dans l'objectif de répondre aux défis énoncés..

1. Order Service :

Il est dédié au domaine de commerce en ligne. Il peut être vu comme notre chargé de relation client étant donné qu'il a pour rôle principal de recevoir les commandes des clients, de faire une demande de livraison à l'entrepôt de stockage et de notifier le client lors de l'arrivée imminente du drone de livraison. De plus, le service Order suit également les états de la commande et de la livraison.
À terme, ce sera à lui de connaître les informations relatives aux clients et au catalogue.

Ce service est l'un de ceux qui devra pouvoir supporter une montée en charge (et descente de charge) la plus critique étant donné que le nombre de commandes effectuées peut varier fortement de façon imprévisible. Un client doit toujours avoir accès à notre service, sans se soucier du nombre d'autres clients qui sont en train de commander en même temps.

Nous avons choisi d'établir une interface de communication avec contrat fort (Remote Procedure Call) pour la réception des commandes à effectuer étant donné qu'il s'agit d'envois critiques et que nous voulons nous assurer du format et de la validité des données que nous recevons. En effet, le temps d'attente de l'utilisateur avant qu'il ne reçoive la confirmation que sa commande est bien enregistrée et prête à être traitée doit être minimal.

2. Warehouse Service :

Il est dédié au domaine de la gestion de colis. Il a pour rôle principal d'être notre buffer des commande : il permet aux techniciens gérant les commandes (représentés par Klaus) de consulter la liste des commandes à préparer pour l'expédition, et de notifier le système une fois la tâche effectuée.

Ce service est découplé de celui qui gère les commandes car il ne suit pas toujours la même temporalité. En effet, une commande effectuée par un Client à un temps donné ne sera pas nécessairement traitée immédiatement, en fonction des contraintes liées au domaine de la gestion des colis : stock manquant, client en livraison prioritaire, etc.
Nous voulons que pour Klaus ces contraintes soient transparentes. Par exemple à terme, si un article n'est pas en stock, la commande ne sera pas présentée à Klaus, mais un processus interne préviendra la personne responsable du réapprovisionnement.

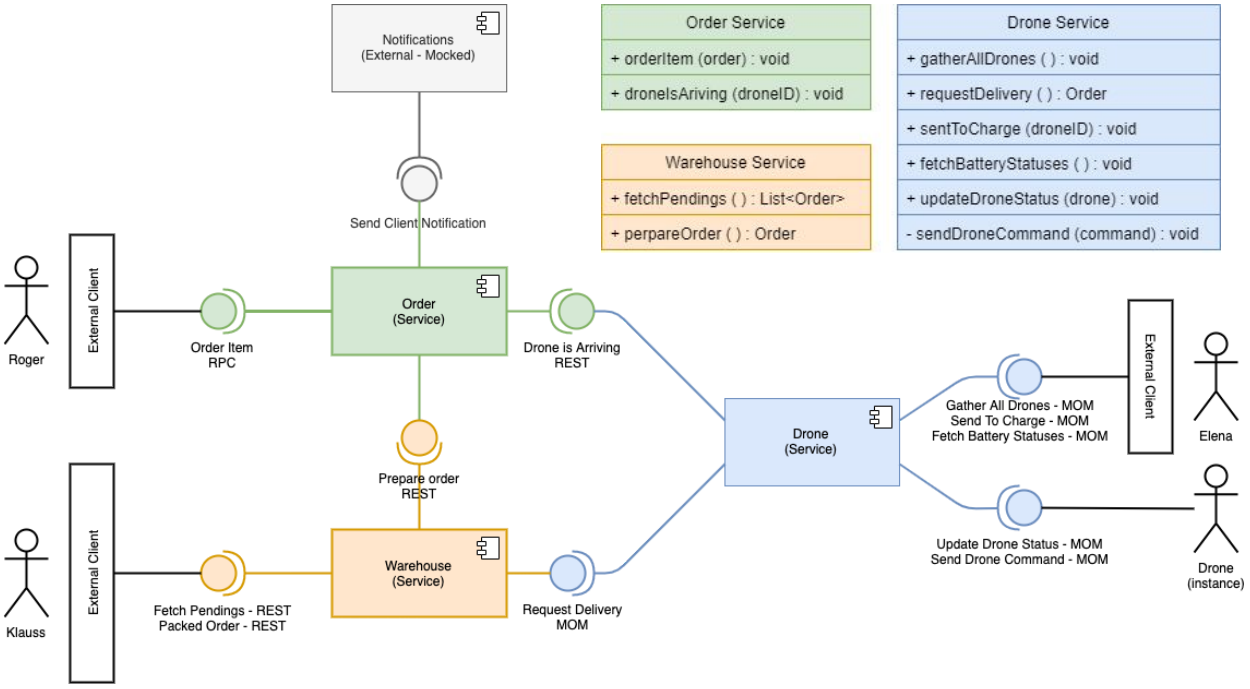
Nous avons choisi d'établir des interfaces de type ressource (REST) pour la réception d'une commande à préparer et pour les interactions de Klaus avec le système étant donné qu'il s'agit de requêtes simples et contenant une version simplifiée de nos Order, ne nécessitant donc pas de contrat fort comme dans le service Order.

3. Drone Service :

Il est dédié à la gestion de la flotte de drones de Dronazone. Il a pour rôle l'affectation d'un drone à l'expédition d'une commande, mais également de permettre à Elena d'avoir une vue d'ensemble sur la flotte.

Ce service est découplé des deux autres, d'une part en raison de son domaine différent : celui de la gestion logistique de livraisons, mais également par sa spécificité technique. En effet, la communication avec les nombreux drones doit se faire par des flux de messages fréquents et nombreux. Nous voulons permettre à Elena d'avoir une vue d'ensemble du statut et du niveau de batterie des drones de la façon la plus approchante du temps réel[1].

Nous avons choisi d'établir des interfaces orientées vers l'envoi de messages et de documents, utilisant des queues de messages pour permettre aux drones de publier à intervalle régulier leur position et leur statut, sans avoir à établir une connection forte à notre système : la perte d'un message de position ou de batterie n'est pas critique puisque l'envoi est prévu toutes les 10 secondes.



Prise de recul

Nous savons que nous allons introduire Kafka dans notre projet prochainement. Nous pensons donc avoir conçu une architecture prenant cela en compte, notamment en utilisant un format déjà proche de l'envoi de messages. Nous savons que nous allons devoir "débrancher" des routes de communication et le rebrancher sur Kafka, mais les données envoyées et les destinataires et émetteurs n'auront pas à changer.

[1] Une communication temps réel avec les drones n'est pas envisageable étant donné qu'ils se trouvent dans un environnement extérieur non contrôlé (perte de réseau temporaire, changement d'antenne réseau, etc). Cela entraînerait par ailleurs une consommation d'énergie qui n'est pas nécessaire selon nous.

Au niveau de l'aspect scalabilité de notre projet, nous pensons que le gros de la charge se concentrera sur le service Order (périodes d'affluences telles que Noël) et dans le service Drone (envoi régulier des données de position et de batterie.) Toutefois, dans le cadre du service Drone il n'y aura pas de montée en charge comme il pourra y en avoir dans le service Order. Nous sommes donc conscients que ce service sera un point critique de notre application et qu'il sera important de le renforcer par la suite.

Diagrammes de séquence

Ce diagramme de séquence correspond au fonctionnement bout-en-bout du cycle de vie d'une commande.

→ Dans un premier temps Roger effectue une commande (traitée par OrderService).

→ Cette commande est transmise à l'entrepôt (via le WarehouseService) afin que Klaus puisse la consulter et la préparer pour la livraison.

→ Une fois prête, Klaus change le statut de la commande et une demande d'expédition est faite (auprès du DroneService)

→ Un drone est alors attribué à la livraison du colis, et va transmettre ses informations (batterie, distance, coordonnées, etc) pour pouvoir être suivi par Elena

→ Une fois le drone à proximité de Roger, le DroneService va demander à OrderService de le notifier. OrderService va lui-même utiliser un service de notification externe pour envoyer le message à Roger.