LE SERVICE DE NOTIFICATION : ÉTUDES D'ARCHITECTURE

1ère étude : *Pour une architecture orientée « microservices »…*

1.0 INTRODUCTION

Le service de notification (« NotificationService »), exposé par le composant (« module », au sens de Maven) du même nom, est au sens logique (ici, le sens de Java) une interface qui fournit (actuellement) au composant de gestion (de la flotte) des drones (et pour autant de sortes de requêtes HTTP) deux méthodes :

- *notifyThatOrderWillShortlyBeDelivered(String orderId)* ;
- *notifyThatOrderWillNotBeDeliveredOnTime(String reason, String orderId)...*

En effet, voici des scénarios qui, mettant en jeu ces (deux) méthodes, me permettent d'implémenter les « user story » 3, 6 et 7, qui définissent, jusqu'à présent, l'ensemble de toutes les « user story » relatives au service de notification.

1.1 Scénario conçu à partir de la « user story » 3 (scénario vert)

(1) Grâce au service de flotte (« FleetService »), exposé par le composant de gestion des drones, le drone chargé de livrer la commande de Roger envoie régulièrement (toutes les secondes, par exemple) ses coordonnées (par une méthode que l'on pourrait nommer « sendCoordinates() » par exemple) au composant de gestion des drones.
(2) Le composant de gestion des drones interroge, par ailleurs, la base de données, afin d'avoir l'adresse de livraison qui connaît l'adresse de livraison (par une requête fondée sur le « getter » de la commande « order.getDeliveryAddress() »).
(3) Puis le composant de gestion des drones calcule, à chaque fois, en « temps réel », le temps restant (en faisant appel à une méthode que l'on pourrait nommer « computeRemainingTime() » par exemple), jusqu'à l'arrivée du drone, au lieu de livraison.
(4) Lorsque la valeur du temps restant est inférieure à une valeur donnée (par exemple, 5 minutes), le composant de gestion des drones demande, au composant de notification, le service de notification (grâce à la méthode « notifyThatOrderWillShortlyBeDelivered() »), afin d'informer Roger du fait que sa commande sera bientôt livrée.
(5) Le composant de notification « fabrique » alors un objet de Notification (grâce à une « NotificationFactory », notamment), à partir de l'ID de la commande : toute notification est datée (à la minute près) et contient un message (qui lui est « attribué » comme une chaîne de caractères) ; ici, le composant de notification a besoin du genre et du nom de Roger, du numéro de la commande (à ne pas confondre avec l'ID) et de l'adresse de livraison (qui peut être distincte de l'adresse (de résidence) principale de Roger).
(6) Une fois que la notification a été « construite », elle est ajoutée à la liste des notifications « de » la commande, puis retournée au composant de gestion des drones (à travers la réponse HTTP).
(7) Enfin, le composant de notification l'envoie à un système externe, chargé de la transmettre à Roger…

1.2 Scénario directement conçu à partir de la « user story » 6 (scénario rouge)

(1) Grâce au service de flotte, exposé par le composant de gestion des drones, le drone chargé de livrer la commande de Roger envoie régulièrement (toutes les secondes, par exemple) ses coordonnées (par une méthode que l'on pourrait nommer « sendCoordinates() » par exemple) au composant de gestion des drones.
(2) Le composant de gestion des drones requête, par ailleurs, régulièrement (toutes les cinq minutes, en temps normal, par exemple) un service externe permettant d'obtenir des données météorologiques (un service de Météo-France, par exemple).

(3) Lorsque le drone chargé de livrer la commande de Roger est menacé par des risques d'intempérie, Elena le rappelle (grâce au service de flotte, qui devrait fournir, logiquement, à cette dernière, une méthode que l'on pourrait nommer « callBack() »).
(4) Le composant de gestion des drones demande alors, au composant de notification, le service de notification (grâce à la méthode « notifyThatOrderWillNotBeDeliveredOnTime() »), afin d'informer Roger du fait que sa commande ne sera pas livrée à temps.
(5) Le composant de notification « fabrique » alors un objet de Notification (grâce à une « NotificationFactory », notamment), à partir de l'ID de la commande, également, mais aussi, et surtout, de la cause, cette fois : toute notification est, encore une fois, datée (toujours « à la minute près ») et contient un message ; ici, le composant de notification a, en plus du genre et du nom de Roger, et du numéro de commande, besoin de la raison de l'abandon, par le drone, de sa mission…
(6) Une fois que la notification a été « construite », elle est ajoutée à la liste des notifications « de » la commande (par une requête fondée sur la méthode de la commande « order.addNotification() »), puis retournée au composant de gestion des drones (à travers la réponse HTTP).
(7) Enfin, le composant de notification l'envoie à un système externe, chargé de la transmettre à Roger…

1.3 Scénario indirectement conçu à partir de la « user story » 6 (scénario orange)

Le 3ème scénario que je propose est une variante du scénario que je viens de décrire ; en effet, au lieu de prendre en considération le cas de risques d'intempérie, j'intègre le cas de la « user story » 4, à savoir le cas d'un niveau de charge de la batterie du drone missionné pour livrer la commande de Roger trop bas :

(1) Grâce au service de flotte, le drone chargé de livrer la commande de Roger envoie régulièrement (toutes les minutes, par exemple) le niveau de charge de sa batterie (par une méthode que l'on pourrait nommer « sendBatteryCharge() » par exemple) au composant de gestion des drones.
(2) Lorsque le drone missionné pour livrer la commande de Roger présente un niveau de charge de sa batterie trop bas, Elena le rappelle (grâce au service de flotte, qui devrait fournir, logiquement, à cette dernière, une méthode que l'on pourrait nommer « callBack() »).
(3) Le composant de gestion des drones demande alors, au composant de notification, le service de notification (grâce à la méthode « notifyThatOrderWillNotBeDeliveredOnTime() »), afin d'informer Roger du fait que sa commande ne sera pas livrée à temps.
(4) Le composant de notification « fabrique » alors un objet de Notification (grâce à une « NotificationFactory », notamment), à partir de l'ID de la commande, également, mais aussi, et surtout, de la cause, cette fois : toute notification est, encore une fois, datée (toujours « à la minute près ») et contient un message ; ici, le composant de notification a, en plus du genre et du nom de Roger, et du numéro de commande, besoin de la raison de l'abandon, par le drone, de sa mission…
(5) Une fois que la notification a été « construite », elle est ajoutée à la liste des notifications « de » la commande (par une requête fondée sur la méthode de la commande « order.addNotification() »), puis retournée au composant de gestion des drones (à travers la réponse HTTP).
(6) Enfin, le composant de notification l'envoie à un système externe, chargé de la transmettre à Roger…

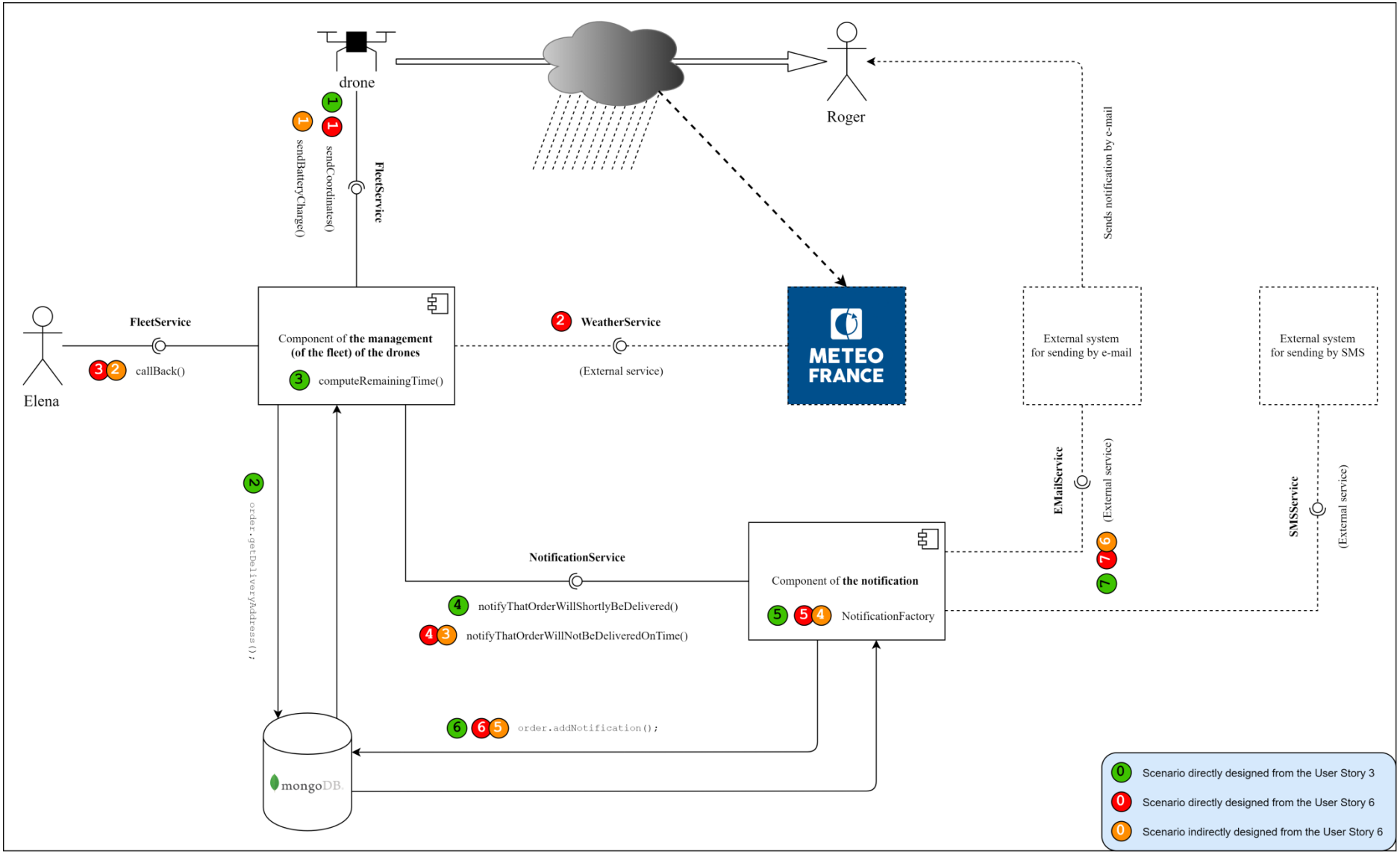

Figure 1 – *Diagramme de l'architecture relative au Service de Notification, traversée par chacun des (trois) scénarios*

1.4 CONCLUSION

Lorsque j'aurai pris en charge la « user story » 7, le composant de notification choisira le système externe (d'envoi) en fonction des préférences (personnelles) de Roger : « Roger, souhaite-t-il recevoir les notifications par e-mail ou SMS (message "texte") ? »

2ème étude : *Pour une architecture orientée « événements »…*

2.0 INTRODUCTION