Projet d'AL
Phase 2

V3: drone autonomy from truck:
allocation for driver, scheduling, mobile possibly disconnected

Team G

Martin Bruel, Thibaut Esteve, David Lebrisse,
Nathan Meulle, Kévin Ushaka

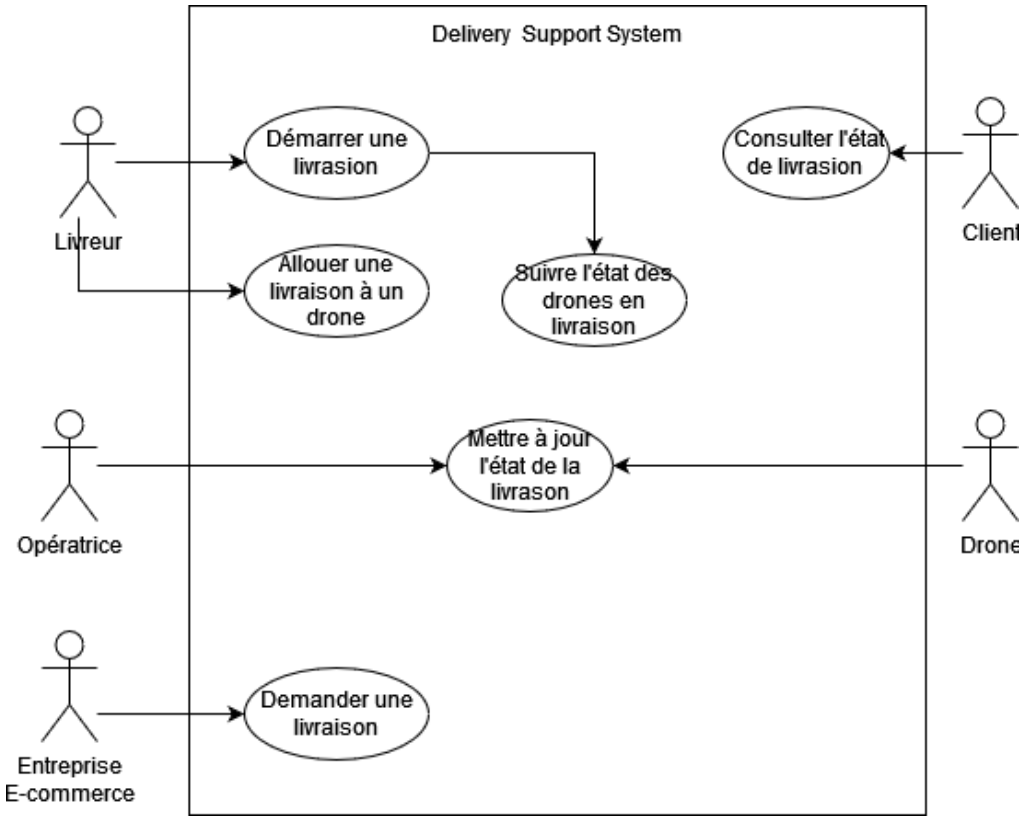
Sommaire

1. Analyse du métier

L'objectif de notre application est la livraison de colis par drone. Nous n'intervenons pas dans la mise en vente des produits et la création de commandes qui sont effectuées par des services externes. De plus, nous nous focaliserons sur l'allocation des drones, la planification des livraisons et la résilience dans la réalisation de la livraison. Dans cette partie, nous analyserons les différents acteurs qui interviennent dans le système et les cas d'usage auxquels nous devons répondre.

a. Cas d'usage

Dans la livraison de colis, nous avons identifié les acteurs suivants : le livreur, le drone, le client et l'opérateur. Le client commande son colis sur une plateforme de e-commerce, il consulte régulièrement l'état de sa livraison. L'entreprise de e-commerce confirme la commande et envoie la livraison à notre entreprise. Les colis sont récupérés et ils sont distribués dans les camions de livraison par les opérateurs qui travaillent dans les différents dépôts. Le livreur, avec son camion, est chargé de déplacer les drones et les colis pour démarrer les livraisons. Sur le terrain il demande au système l'allocation des drones et démarre les livraisons. Le système prend le relais et suit l'état des drones. Les drones, eux, livrent les colis en respectant un itinéraire.



b. Personas

Sam est un client. C’est un retraité de Genouillac. Il commande souvent des produits depuis des sites de e-commerces car aucune grande ville n'est à côté de sa maison et qu’il ne peut pas se déplacer facilement. Il souhaite connaître le statut de la livraison pour être prêt à le recevoir à sa porte.

Christine est opératrice de traitement de colis. Elle travaille dans un dépôt pour une société de livraison de colis par drones. Elle est chargée de récupérer et redistribuer les colis qu’elle reçoit. Elle veut charger les camions qui partent pour livraison avec le colis des clients.

Marcel est livreur pour une agence de livraison à domicile. Sa société se spécialise dans la livraison par drones. Il habite à Genouillac dans la Creuse. Sa région est très faiblement peuplée et très peu couverte par le réseau 4G ou 5G. Il est chargé de livrer chaque matin les colis qui sont déposés dans son camion. Il souhaite réaliser facilement toutes ses livraisons avec les drones qui lui sont fournis par sa société. Une fois arrivé au point de déploiement, il est chargé de la synchronisation des drones avec le camion puis doit se tenir prêt à intervenir en cas de problème.

2. ***Roadmap (phase 2)***

a. ***Scénarios***

Scénario 1 [Obligatoire]: Livraison de plusieurs colis en flotte

- Le conducteur arrive sur le lieu de déploiement des drones et démarre sa tablette.
- Le conducteur récupère une flotte de drones livrant des colis dans une même zone
- Le conducteur charge les colis sur chacun des drones de la flotte et indique que les colis sont chargés sur la tablette.
- Le camion communique aux drones l'itinéraire de livraison et un drone référent (drone leader).
- Chaque drone signale au camion qu'il est parti pour la livraison.
- Au cours du trajet, le camion reçoit les positions des drones.
- Le drone leader arrive à l'adresse avec sa flotte et chaque drone dépose son colis.
- Les drones notifient le camion que les colis ont été déposés.
- La flotte repart pour rejoindre le camion.
- Le camion signale que les colis sont livrés dès qu'il peut se connecter à internet.
- Les drones signalent qu'ils sont retournés sur le camion et qu'ils sont disponibles pour une nouvelle livraison.
- Le conducteur redemande les assignations des colis et drones.
- Le système recalcule l'allocation des colis et assigne des colis à une nouvelle flotte de drones qui vient de revenir.

Scénario 2 [Obligatoire]: Un drone de la flotte n'arrive plus à contacter le camion

- Les drones signalent qu'ils sont partis pour les livraisons et commencent la navigation.
- Durant le vol, un des drones n'arrive plus à communiquer avec le camion. Il contacte alors le drone leader.
- Le drone leader sert de relais et transfère les informations du drone au camion.
- La flotte continue son itinéraire jusqu'à la destination

Scénario 3 [Obligatoire]: Dispatch de plusieurs livraisons depuis un drone leader

- Le conducteur arrive sur le lieu de déploiement des drones et démarre sa tablette.
- Le conducteur récupère une flotte de drones livrant des colis dans une même zone
- Le conducteur charge les colis sur chacun des drones de la flotte et indique que les colis sont chargés sur la tablette.

- Le camion communique aux drones l'itinéraire de livraison et un drone référent (drone leader).
- Le drone leader signale qu'il est parti avec sa flotte pour les livraisons et commence la navigation.
- Arrivé à destination, le drone leader dispatch chaque drone vers leurs adresses de livraison
- Chaque drone arrive à destination et signale qu'ils ont fini leur livraison
- Chaque drone revient au drone leader
- La flotte repart pour rejoindre le camion. cf Scénario 1

Scénario 4 [Obligatoire]: Election d'un nouveau leader et création d'une deuxième flotte

- Arrivé à destination, le drone leader dispatch chaque drone vers leurs adresses de livraison
- Le drone leader s'éloigne d'une partie de la flotte pour réaliser sa livraison
- Une partie de la flotte ne peut plus se connecter au leader, ils élisent un nouveau drone leader
- Chaque drone arrive à destination et signale qu'ils ont fini leur livraison
- Chaque drone revient au drone leader
- La flotte repart pour rejoindre le camion. cf Scénario 1

Scénario 5 [Obligatoire]: Connection perdue entre le camion et le drone leader

- Le drone leader n'arrive pas à contacter le camion pour sur la position de sa flotte (demandée à intervalle régulier).
- Le camion enregistre la dernière position connue du drone leader et lance un timeout (calculé en fonction de la distance à parcourir). Une fois ce délai dépassé, le groupe est considéré comme perdu (la dernière position connue sera envoyée au site pour permettre l'envoi d'une équipe de récupération).
- Le groupe poursuit son plan de vol. Il effectue une synchronisation lors de sa reconnexion.

b. Features

Epic Planification

En tant que camion
Je veux assigner une flotte de drones pour une livraison
Afin de livrer plusieurs colis dans une zone

En tant que camion
Je veux choisir le drone leader d'une flotte
Afin de pouvoir communiquer avec les drones du groupe

Epic Livraison

En tant que drone follower
Je veux pouvoir signaler avoir livré mon colis au drone leader
Afin d'assurer un suivi du colis

En tant que drone leader
Je veux dispatch les drones follower quand j'arrive dans la zone de livraison
Afin de réaliser plusieurs livraisons dans une même zone

Epic Suivi/Pistage

En tant que drone follower
Je veux communiquer ma position au drone leader
Afin d'assurer l'avancée de la flotte

En tant que drone leader
Je veux détecter qu'un drone ne réponds plus
Afin de pouvoir lancer la recherche

En tant que drone leader
Je veux chercher un drone perdu avec les drones follower
Afin de retrouver le drone perdu

En tant que drone leader
Je veux pouvoir communiquer via un drone intermédiaire
Afin de communiquer à un drone éloigné

En tant que drone follower
Je veux voler à proximité des autres drones de la flotte
Afin de rester en groupe

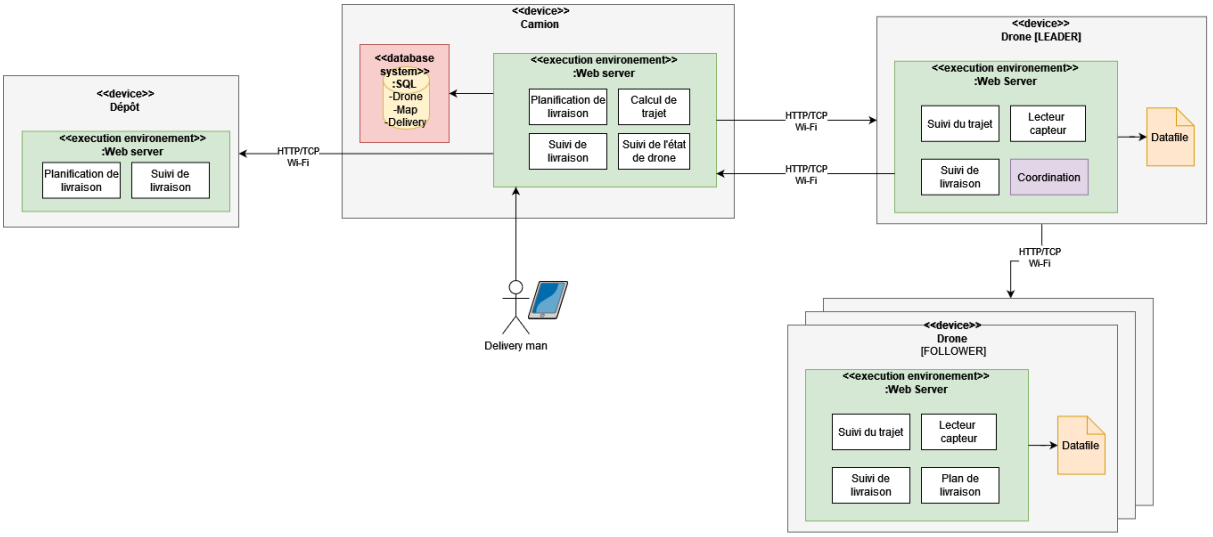
Epic Notification

En tant que camion
Je veux récupérer les informations d'une flotte
Afin de suivre l'avancement de la livraison

3. Architecture

a. Architecture système (Rappel)

(cf : https://github.com/pns-si5-al-course/al-drone-21-22-al-drone-21-22-g/blob/main/archi3.png)
Notre système propose la livraison de colis par drone à partir d'un entrepôt. Nous avons découpé notre architecture en trois parties : l'entrepôt, le camion et les drones.



L'entrepôt est assimilé à un service externe que nous avons mocké.

Le camion est représenté par un monolithe n-tiers. Celui-ci dispose d'une application sur un serveur local qui planifie les livraisons, calcul les trajets et le suit l'état des livraisons. Ce serveur stocke les données de livraison, les informations sur les drones, et les cartes géographiques de la région. De cette manière, il est plus facile de monitorer le système avec les autres services (drones, warehouse),et nous n'avons pas de besoin de scaling car le nombre de drones et de colis sont limité.
Quand le livreur arrive à son point d'arrivée, il se connecte au point de connexion du camion et appelle l'application pour démarrer la planification des trajets et l'assignation des drones. Les drones sont connectés en *filaire* quand ils sont sur le camion. Le livreur récupère les différentes assignations et les informations sur la livraison. Il charge le drone. Après le chargement, le livreur débranche le drone et confirme le démarrage de la livraison. Le drone se connecte en Wifi au camion et démarre la navigation. Dans la phase

2 du projet, le livreur pourra également démarrer une flotte de drones pour livrer plusieurs colis dans une zone.

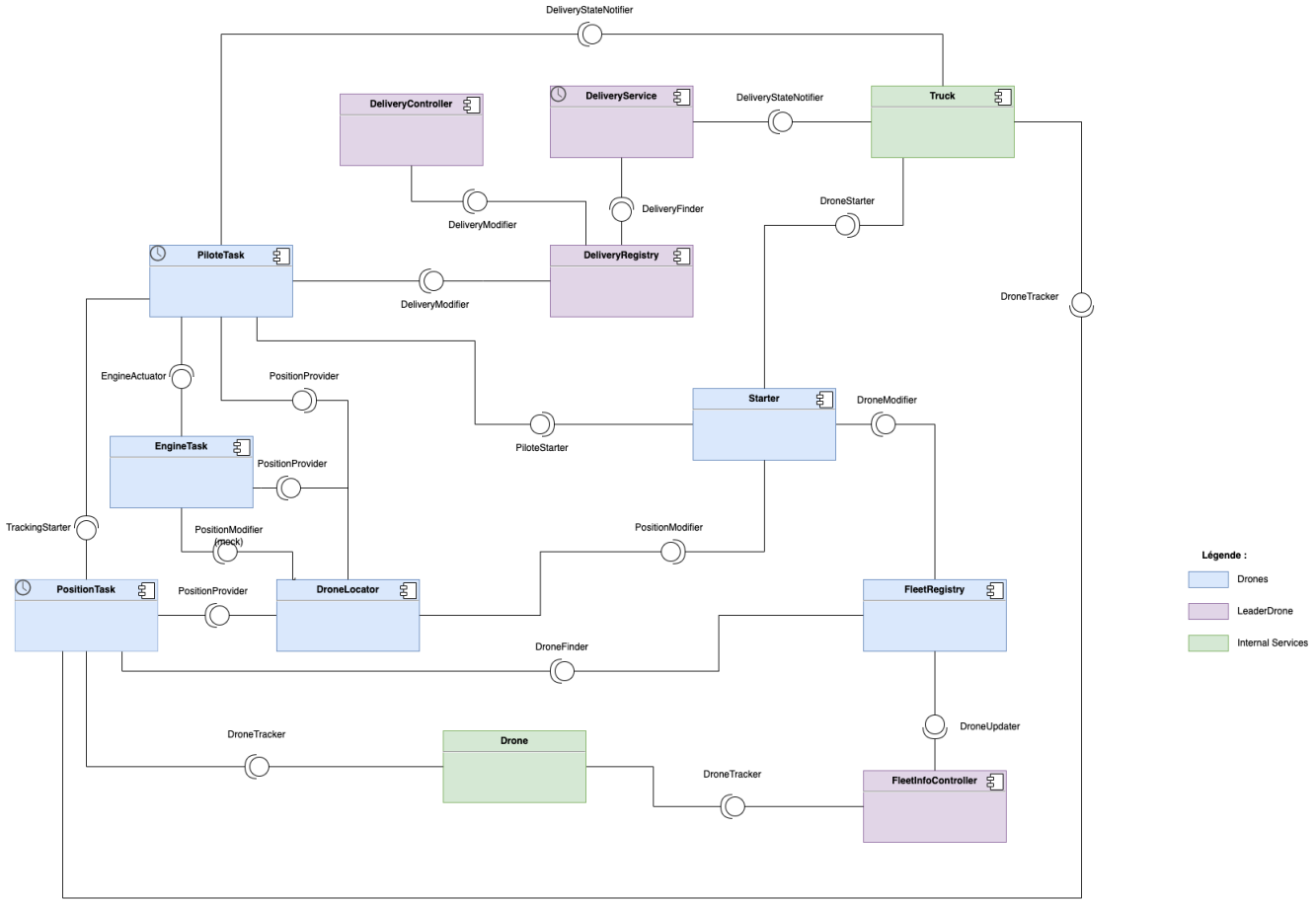
Le camion demande régulièrement l'état du drone ou du drone leader d'une flotte. En prenant en compte la possibilité d'une perte de connexion, l'on requiert plusieurs fois le drone et mettons en place un timeout avant de définir le drone ou la flotte comme perdu. De son côté, le drone enregistre dans sa base de données ses dernières positions pour éventuellement revenir à son dernier point de livraison en cas d'obstacle. Il a également les informations de la livraison pour mettre à jour son état.
En effet, le camion orchestre tout et demande les communications cela permet d'éviter l'aspect événementiel entre le camion et le drone qui aurait été plus fastidieux.

Le drone gère des tâches en concurrence : il lit les données de ses capteurs, il les traite, suit le trajet et envoie son état au camion. Quand il est drone leader, celui-ci suit également la progression des autres drones de sa flotte et s'assure que toutes les machines soient présentes. Quand le drone revient au camion, on le branche pour le recharger et lui transférer sa prochaine livraison. L'état des livraisons sur le serveur du camion est également mis à jour. Le retour sur camion à l'entrepôt synchronise les données de livraison avec les données du serveur de l'entreprise.

b. ***Diagramme de composants***

Nous conservons le précédent diagramme de composants pour la partie du camion (cf : https://github.com/pns-si5-al-course/al-drone-21-22-al-drone-21-22-g/blob/main/component4.png).

Voici le nouveau diagramme développant les composants du drone :
(cf : https://github.com/pns-si5-al-course/al-drone-21-22-al-drone-21-22-g/blob/main/componentDrone2.png)



StarterController : reçoit les informations au démarrage du camion

PiloteTask : envoie le prochain cap à l'EngineTask à partir de l'itinéraire

EngineTask : récupère la position et fait actionne les moteurs du drone pour le faire avancer dans la bonne direction

EngineTaskFake : implementation fake de l'EngineTask qui modifie activement la position reçue par le capteur : DroneLocator

PositionTask (follower uniquement) : consulte la position à intervalle régulier et émet vers le camion ou vers le drone leader via FleetInfoController

SearchService :

- leader : contacte tous les followers pour retrouver le drone A
- follower : tente de contacter le drone A. En cas de succès : envoi de sa position.

DeliveryService (leader uniquement) : envoie les informations de livraison au camionde manière asynchrone (mécanisme de retry)

DroneLocator : lit les capteurs pour retourner la position du drone

FleetInfoController : point d'entrée pour les drones follower afin de transmettre leurs positions

c. ***Choix d'architecture***

- Lors de notre première approche, nous avons essayé d'établir des algorithmes de communications entre des drones distincts uniquement par leur id, ce qui à introduit la notion de cycles.
  Afin de simplifier les communications au sein d'une flotte nous avons choisi d'instaurer le concept leader/follower. Les drones dit "follower" communiquent leurs informations au leader qui les communique à son tour au camion. Cela permet, lors de l'entrée en zone blanche, que tous les drones aient un référent.
- Les drones follower communiquent directement avec le camion lors de la phase d'atterrissage et de décollage. En effet, on considère que lors de ces deux phases les drones sont juste à côté du camion donc pas en zone blanche. Cela nous permet également de récupérer des briques de code déjà implémentées.
- Des drones follower ne peuvent pas rester en zone blanche tout seul. Si un drone follower n'arrive pas à contacter le leader, il effectue un vol stationnaire au bout de 10 tentatives de 10 secondes.
  Si un drone leader se rend compte qu'il à perdu un drone, il va effectuer une recherche parmi les drones de la flotte pour qu'ils vérifient s'ils peuvent contacter le drone recherché. Une recherche avec succès par un drone réinitialise le compteur de tentative du drone leader.
- Nous avons simuler le déplacement du drone en partant du principe que nous utilisons une interface bas niveau faisant déplacer le drone. Ainsi nous avons mocker les actionneur via le composant Engine (déplacement du drone) et les capteur par DroneLocator (position actuel du drone). Cela nous permettra de déployer notre solution facilement sur de vrai plateforme

Interfaces entre le camion et les drones :

DroneStarter

```
POST <drone_ip>/drone-api/delivery/start
JSON Body : { flightPlan : List<Position>,
              fleet : List<Drone>,
              leaderId: long }
with Drone : { id: long, ip: string }
```

DroneTracker

```
POST <truck_ip>/truck-api/position
POST <drone_ip>/drone-api/position
JSON Body : [
              {
                droneId: long,
                position: Position,
                timestamp: long
              }
             ]
```

DeliveryStateNotifier

```
POST <truck_ip>/truck-api/delivery/
JSON Body : { droneId: long, deliveryState: int}
```

DeliveryModifier

```
packageDelivered( packageId: int) : void
```

PiloteStarter

```
startJourney( flightPlan : List<Position>) : void
```

EngineActuator

```
flyTo( position: Position) : void
```

PositionModifier

```
setCurrentPosition( newPosition: Position ) : void
setFollowerPosition(newPosition : Position, drone : Drone) : void
```

TrackingStarter

```
startSendingPositions() : void
```

PositionProvider

```
getCurrentPosition() : Position
```

DroneFinder

```
findAll() : List<DroneInfo>
findLeader() : DroneInfo
findDroneDisconnected(): List<Drone>
with DroneInfo : { drone : Drone, position : Position}
```

DroneModifier

```
registerFleet(fleet : list<Drone>, fleetLeader: Drone) : void
with Drone : { id: int, isLeader: boolean, packageId: int }
```

DroneUpdater

```
modifyPositionDrone( droneId: long, position: Position,
timestamp: long)
```