Projet d’AL

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

Analyse du métier **3**
- Cas d’usage 3
- Personas 4

Roadmap **5**
- Scénarios 5
- Epics 7
- Organisation de l'équipe 8
- Sprints 8

Analyse fonctionnelle **11**
- Architecture système 11
- Architecture application 13
  - Diagramme de composants 13
  - Justifications 13
  - Présentation des composants 14
  - Présentation des interfaces 15
  - Déroulement du scénario 1 dans notre architecture 18
  - Modèle de données 20
  - Evolution de l’architecture 20
- Architecture technique 21

Rétrospective **22**

Auto Évaluation **23**

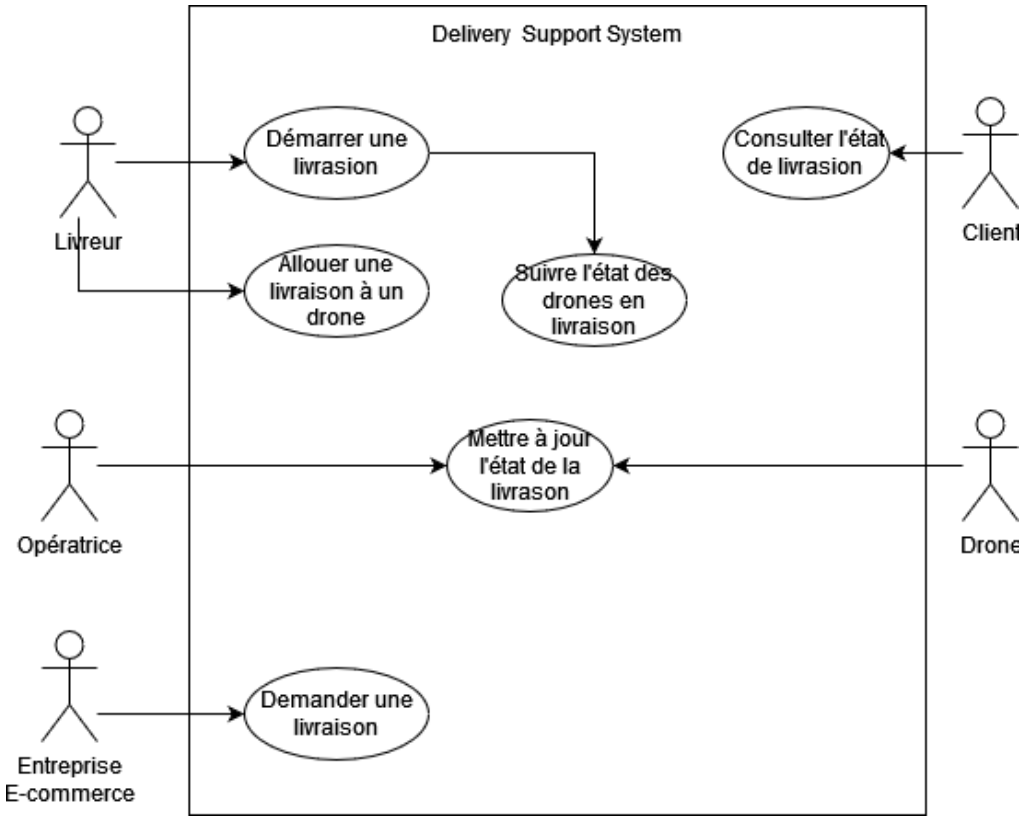
Bibliographie **24**

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***

a. ***Scénarios***

Scénario 1 [Obligatoire]: Chaque drone livre un colis à la fois

- Le conducteur arrive sur le lieu de déploiement des drones et démarre sa tablette.
- Le conducteur lit sur sa tablette le colis qu'il doit associer au drone libre.
- Il charge le colis sur le drone.
- Le conducteur indique que le colis est chargé sur la tablette.
- Le camion communique au drone l'itinéraire de livraison.
- Le drone signale qu'il est parti pour sa livraison et commence la navigation.
- Il arrive à l'adresse et dépose le colis, il n'est alors pas connecté au camion
- Il repart pour rejoindre le camion.
- Arrivé dans la zone de couverture du camion, il signale qu'il a livré le colis.
- Le camion signale que le colis est livré dès qu'il peut se connecter à internet
- Le drone signale qu'il est retourné sur le camion et qu'il est libre
- Le conducteur redemande les assignations des colis et drones
- Le système recalcule l'allocation des colis et assigne un colis au drone qui vient de revenir

Scénario 2 [Obligatoire]: Un grand drone livre plusieurs colis ou un seul.

- Le camion comporte des drones classiques capables de soulever 1 colis léger et des drones lourds pouvant supporter 1 colis lourd ou plusieurs colis légers.
- Arrivé sur le site de livraison, le conducteur lit sur sa tablette le colis qu'il doit associer au drone libre.
- Le conducteur a un colis lourd en priorité et d'autres colis légers qui doivent être livrés ensuite.
- Le conducteur charge le colis lourd et envoie le drone effectuer sa livraison (cf Scénario 1).
- Le drone revient, il ne reste que des colis légers. Le conducteur charge tous les colis dans ce drone et l'envoie pour sa dernière livraison.

(La distinction colis léger/lourd n'a pas été implémentée contrairement à celle petit/grand drone capable de livrer plusieurs colis.)

Scénario 3 [Obligatoire]: Le drone ne revient pas au camion

- Le drone est parti pour sa livraison.
- Il échange sa position avec le camion.
- Le camion ne reçoit plus de données du drone.
- Le camion détermine un timeout en fonction de la distance qui reste à parcourir par le drone.
- Le timeout est terminé, le camion considère que le drone est inopérant.

- Le système réalloue les colis aux drones qu'il reste en fonction de leur taille.
- Le camion signale les colis qui ne pourront pas être livrés

Scénario 4 [Obligatoire]: Route accidentée

- Le camion ne peut pas atteindre la position prévue.
- Le chauffeur s'en approche le plus possible.
- Il relance le calcul des plans de vols à partir de sa nouvelle position.
- Une livraison n'est pas réalisable (beaucoup trop loin), elle est annulée.
- Le conducteur lance la livraison des autres colis avec les drones.

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

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

Scénario 6 [Optionnel]: Le drone rencontre un obstacle

- Le drone est parti pour sa livraison.
- Il échange sa position avec le camion.
- Il rencontre un obstacle (entraînement des pompiers, girafe à long cou, migration d'oiseaux, cumulonimbus,...).
- Il revient au dernier point de connexion avec le camion.
- Il envoie sa position au camion et demande un recalcul du plan de vol.
- Le camion calcule et renvoie un premier petit itinéraire alternatif.
- Le drone met à jour son plan de vol. Cela lui permet de corriger sa trajectoire et obtenir les dernières informations quant aux imprévus (météo, commande annulée...)
- En cas d'impossibilité à franchir l'obstacle, le drone revient pour récupérer un tout autre itinéraire.

Scénario 7 [Optionnel]: Un drone n'est pas en mesure de décoller

- Le conducteur lit sur sa tablette le gros colis qu'il doit associer au gros drone libre.
- Il charge le gros colis sur le gros drone.
- Le conducteur indique que le colis est chargé sur la tablette.
- Le camion lui communique l'itinéraire de livraison.
- Le drone ne répond pas, la batterie est défectueuse et n'accepte plus la charge
- Le système réalloue les colis aux drones qu'il reste en fonction de leur taille

- Le camion signale les colis qui ne pourront pas être livrés

Scénario 8 [Optionnel]: Couverture étendue : le camion effectue plusieurs livraisons

- Le conducteur lit sur sa tablette la première position où doit se rendre le camion
- Le conducteur indique sur sa tablette qu'il est arrivé à la première position.
- Une fois arrivé, il déploie ses drônes, ceux-ci vont livrer les colis dans un périmètre autour du camion. (*taille du périmètre à déterminer*)(itération du scénario 1).
- Quand les drônes ont fini leurs livraisons et sont revenus au camion, le conducteur lit sur sa tablette la seconde position où doit se rendre le camion.

Remarque : les coordonnées indiquées au camion sont calculées de manière à optimiser la vitesse de livraison des colis ainsi que la qualité du signal entre le camion et les drones.

Scénario 9 [Optionnel]: Meilleure zone de connection pour le camion

- Au cours de son vol, un drone trouve une meilleure zone de connexion pour le camion.
- Le drone notifie le camion des données de connexion
- Le camion choisit de se déplacer à la zone indiquée
- Le camion notifie chaque drone de sa nouvelle position

b. ***Epics***

En tant que Client,
Je veux que ma commande soit livrée
Afin de pouvoir la récupérer
=> Livraison

En tant que Conducteur,
Je veux savoir comment répartir les drones et colis
Afin d'assurer les livraisons
=> Répartition / Planning

En tant que Camion,
Je veux pouvoir suivre mes drones
Afin de pouvoir déterminer lesquels sont disponibles
=> Suivi / Pistage

En tant que Livreur
Je veux pouvoir optimiser l'utilisation des drones
Afin de livrer plus et plus rapidement
=> Optimisation

c. ***Organisation de l'équipe***

Dans l'équipe, nous avions défini des axes sur lesquels chaque membre s'est plus focalisé mis à part l'architecture et les choix de conception où tout le groupe s'est investi de la même façon. En effet, tout au long du projet nous avons confronté chacun nos design d'architecture par rapport aux scénarios et avons rassemblé les composants intéressants de chaque proposition. A chaque sprint, nous discutions du changement d'interfaces et des composants dont nous avions besoin ou qui étaient inutiles.

Les axes du projets que nous avons défini sont la mise en place de Spring et de JPA, la reconnexion des drones avec le camion, le démarrage du drone et la mise en place de docker, les notifications au dépôt, et la mise en place de la chaîne d'intégration. La mise en place de Spring et de JPA a été réalisée en grande partie par David Lebrisse. Kevin a davantage travaillé sur la reconnexion des drônes avec le camion, Martin Bruel sur le démarrage du drone et la mise en place de docker, Thibaut Esteve sur les notifications résilientes au dépôt et Nathan Meulle sur la mise en place de la CI.

Sur Github nous utilisions une stratégie de branche Gitflow. Nos branches de développement et de release étaient stables et les fonctionnalités seules étaient sur des branches à part. Les merge requests étaient obligatoires pour que chaque membre de l'équipe teste la fonctionnalité et vérifient les changements du code. De cette manière, chaque membre du groupe avait une vision large du projet et de ce qui avait été fait dans les composants sur lesquels il n'avait pas travaillé.

d. ***Sprints***

Le premier sprint était la livraison de plusieurs colis, chaque drone possédant un colis, car cela représente le produit minimal viable et traverse tous les services et composants que nous présenterons plus en détail dans la prochaine partie. Le sprint devait être fini le 12 Octobre mais nous avons pris un peu de retard et l'avons rendu le 14 Octobre. Cela n'a pas eu de grand impact sur notre avancée puisque nous avons réorganisé les sprints pour en faire un de moins sans enlever de fonctionnalités.

- En tant que livreur, je veux consulter la liste d'affectation des colis afin de recharger les drones et débuter la livraison.
- En tant que camion, je veux affecter le colis le plus prioritaire au prochain drone libre afin d'optimiser mes horaires de livraison.(le premier enregistré)
- ~~En tant que drone, je veux télécharger l'itinéraire de livraison afin de démarrer la navigation.~~

  (Cette user story a été supprimée car nous nous partons de l'hypothèse que cette étape a déjà été réalisée avant que le camion parte sur le terrain afin de se

concentrer sur le plus important : la planification et le suivi des livraisons réalisés par les drones.)

- En tant que drone, je veux suivre l'itinéraire de livraison afin de déposer le colis à l'adresse du client et retourner au camion.
- En tant que livreur, je veux que le camion me notifie d'un drone libre et du prochain colis à livrer afin de démarrer une autre livraison.
- En tant que camion, je veux être notifié régulièrement de l'activité des drones, afin de maintenir mon planning de livraison.
- En tant que camion, je veux notifier le dépôt que la livraison est effectuée afin de notifier le client.
- En tant que développeur, je veux que le drone notifie le camion de sa fin de livraison afin de changer l'état de la livraison.

Dans ce deuxième sprint nous nous sommes penchés sur la communication entre le drone et le camion pour commencer à apporter de la valeur par rapport à notre variant. Le drone peut subir une déconnexion voire être perdu. Ce sprint était prévu pour le 19 Octobre et a été réalisé dans les temps. Au bout de ce sprint nous avions déjà une première démonstration qui traite l'une des problématiques de notre sujet.

- En tant que camion, je veux réessayer de me connecter au drone perdu afin de récupérer sa dernière position.
- En tant que camion, je ne peux pas accéder au point de déploiement et indique donc la position alternative trouvée.
- ~~En tant que développeur, je veux annuler les livraisons qui sont beaucoup trop éloignées du camion afin de réussir mes livraisons.~~
  (Cette fonctionnalité a été reportée au dernier sprint car elle n'est pas dans le thème de ce sprint qui est autour de la perte de connexion avec le drone.)

- En tant que développeur, je veux ré assigner les colis rattachés au drone qui est perdu afin de garantir les livraisons restantes dans les meilleurs délais.

Dans le dernier sprint, nous nous sommes concentrés sur l'optimisation des itinéraires de livraisons afin de complexifier la planification et l'allocation des drones.

- En tant que camion, je veux affecter les colis en fonction de leur poids afin de ne pas surcharger les drones.
- En tant que camion, je veux pouvoir associer les gros drones à plusieurs colis afin de livrer plus et plus rapidement

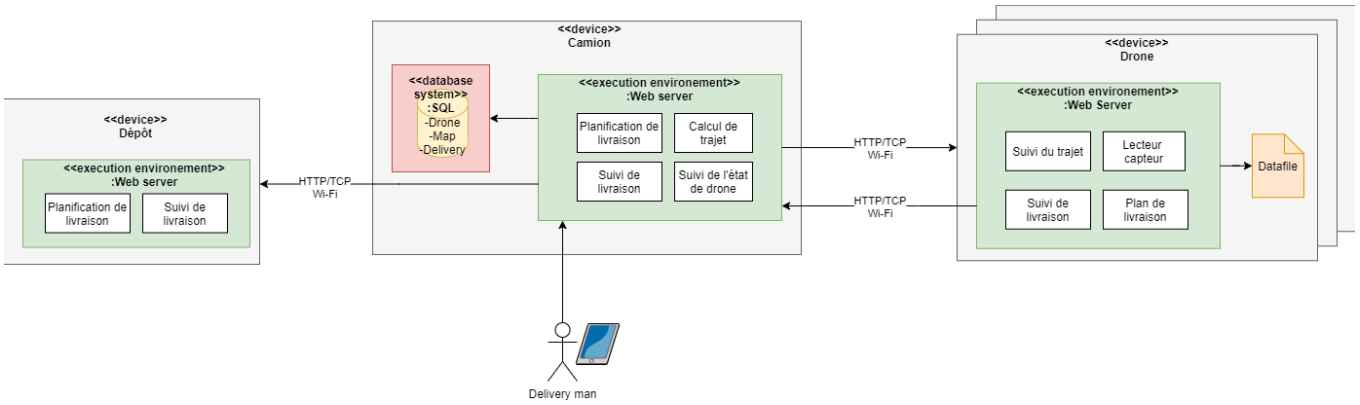
- ~~En tant que camion, je veux déterminer une nouvelle zone de décollage afin d'optimiser mon plan de route~~.
  (Cette user story est devenue une fonctionnalité dans nos perspectives futures qu'on implémenterait avec le contournement d'obstacles avec les drones.)

- En tant que livreur, je veux lancer le calcul des plans de vols à partir d'une nouvelle position afin de m'adapter aux accidents de la route.
- En tant que développeur, je veux annuler les livraisons qui sont beaucoup trop éloignées du camion afin de réussir mes livraisons.

3. Analyse fonctionnelle

a. Architecture système

(cf : https://github.com/pns-si5-al-course/al-drone-21-22-al-drone-21-22-g/blob/main/archi2.jpg)



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.

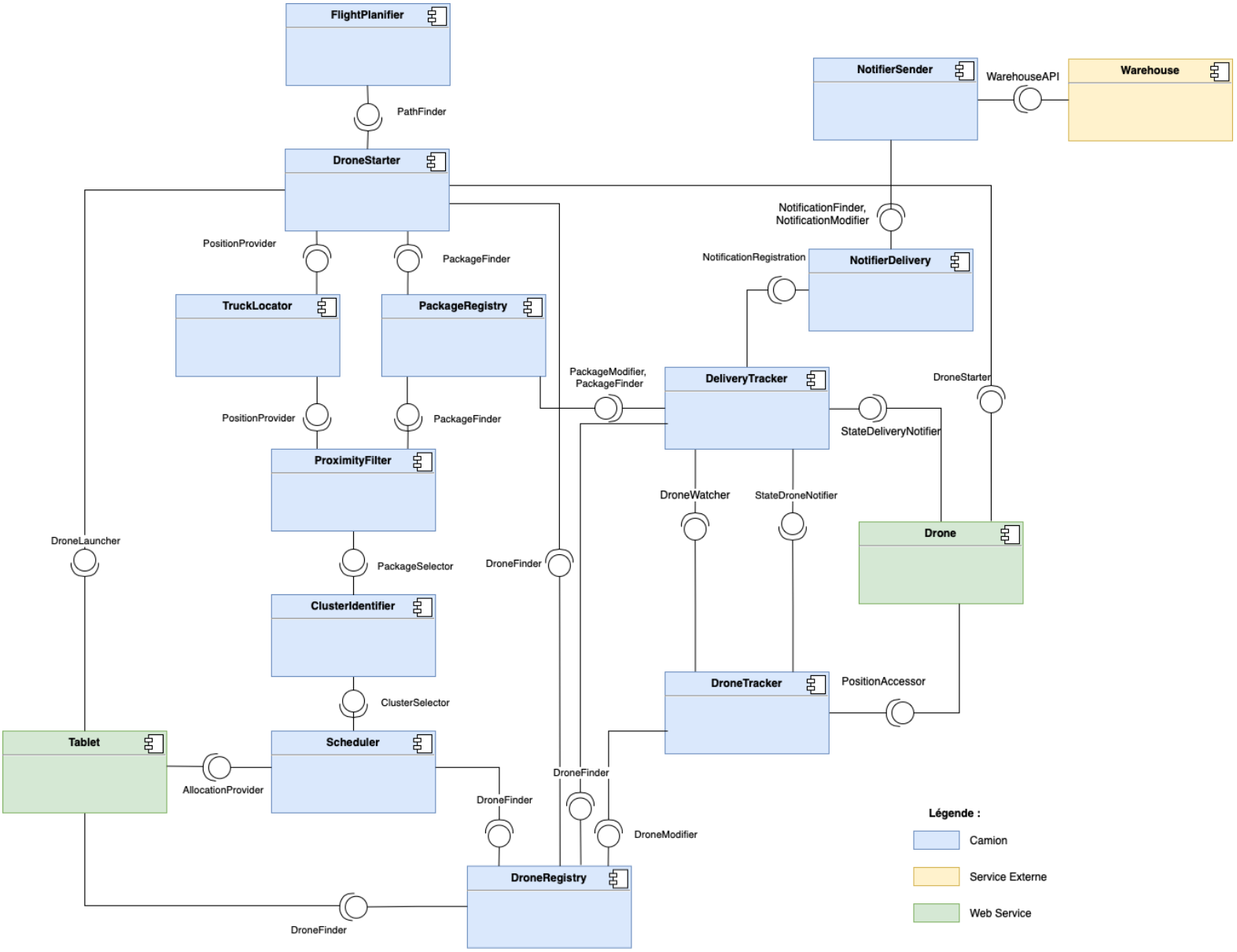
Le camion est représenté par un monolithe n-tiers. Celui-ci dispose d'une application sur un serveur local qui planifie les livraisons, calcul les trajets et le suit l'état des livraisons. Ce serveur stocke les données de livraison, les informations sur les drones, et les cartes géographiques de la région. 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.

Le camion demande régulièrement l'état du drone. 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 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. 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 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. *Architecture application*

Diagramme de composants



Justifications

Notre système doit pouvoir couvrir trois principales problématiques liées au sujet sélectionné. Chaque problématique ayant des spécificités bien distinctes et pouvant évoluer dans l'avenir, chacune des parties du code permettant de les traiter doit être bien isolée. En effet, chaque changement aura un impact sur des composants bien déterminés. De plus, cette division nous permet de réaliser des tests plus fins sur des cas d'usage précis et nous permet d'identifier simplement lors du développement ce que

traite chacun des composants. Ainsi chacune des problématiques est traitées de la façon suivante :

- Perte de connexion

Cette problématique implique que notre système soit dans la capacité de communiquer avec les différentes entités en fonction de l'état de la connexion. Les composants *NotifyScheduler* et *DroneTracker* certifient que les communications avec ces entités se font bien. Sinon un mécanisme de "retry" est mis en place pour renvoyer la requête en cas d'échec. De plus, notre système doit pouvoir savoir si un drone ne répond pas à cause du réseau ou à cause du fait qu'il est perdu. *DroneTracker* intègre une comptabilisation du nombre de "retry" pour détecter les drones perdus.

- Livraison de plusieurs colis par un même drone

Cette problématique implique une association plus fine des paquets avec les drones. Les composants *ClusterIdentifier* et *Scheduler* vont permettre d'optimiser les associations notamment en détectant les colis ayant une adresse de livraison proche et en donnant la priorité aux gros drones par rapport aux petits drones. De plus, le *FlightPlanifier* est dédié à la détermination de l'itinéraire en prenant en compte les différentes livraisons d'un gros drone.

- Intervention dans une zone sinistrée

Cette problématique implique le fait que le camion ne puisse pas rejoindre l'emplacement initialement déterminé au départ du dépôt par exemple si un arbre est tombé sur sa route. Les composants *TruckLocator* et *ProximityFilter* doivent pouvoir dynamiquement sélectionner les colis livrables dans le rayon d'action du camion.

Présentation des composants

Nous n'avons pas détaillé le fonctionnement interne du drone et du système de l'entrepôt. Nous faisons ici l'hypothèse que l'entrepôt dispose déjà d'une solution présentant une API dont nous faisons l'usage. Quant à la tablette, elle est modélisée par une CLI à travers laquelle le conducteur effectue ses actions sur le système. C'est pour cela que nous allons essentiellement décrire les composants qui s'exécutent sur le serveur du camion.

Le livreur commence sa livraison en démarrant la planification. C'est le *Scheduler* qui planifie les livraisons, il récupère les colis et cherche des drones disponibles pour la livraison. Le composant *ClusterIdentifier* permet de déterminer des colis proches pouvant être livrés par un même gros drone. Celui-ci récupère les colis depuis *ProximityFilter* qui détermine ceux pouvant être livrés dans le rayon d'action du camion. *DroneRegistry* permet de trouver les drones disponibles et le *PackageRegistry* permet de trouver les colis à livrer. Le *Scheduler* associe les colis aux drones et renvoie une liste d'assignation.

Lorsque le livreur termine de charger le paquet sur le drone, il demande ensuite le démarrage du drone. L'application doit alors calculer le trajet que suivra le drone grâce à la position récupérée du camion et l'adresse de destination. En effet, notre composant *DroneStarter* recevra de la tablette l'identifiant du colis et du drone et communiquera avec *TruckLocator* et *PackageRegistry* afin de récupérer les positions de départ et d'arrivée. Il passe les informations à *FlightPlanifier* qui calcule un itinéraire à partir de positions GPS.

Le drone étant lancé, il n'est plus disponible pour les prochaines livraisons. Pour prendre celà en compte, lors du décollage du drone, ce dernier confirme le début de livraison auprès de *DeliveryTracker* et l'on modifie son statut de livraison ce qui nous permet de le retirer des drones disponibles dans *DroneRegistry*.

Cependant, il arrive que les drones ne puissent pas revenir. Pour gérer cette éventualité, lors du 'message' de décollage, on active un composant, *DroneTracker*, chargé de traquer ce drone, afin de s'assurer de son état de fonctionnement. Au retour du drone, on arrêtera de le traquer. Un drone perdu se traduit par une livraison échouée, c'est pourquoi un composant, *DeliveryTracker*, met à jour l'état de la livraison et notifie le dépôt de l'état de livraison.

Présentation des interfaces

AllocationProvider

```
GET /truck/allocation
JSON Body : Allocation[]
```

DroneLauncher

```
POST /start/drone/:droneId/package/:packageId
JSON Body : none
```

PositionProvider

```
getTruckPosition() : Position
```

PackageFinder

```
getDeliverablePackages() : List<Delivery>
getPackageByPackageId(packageId long) : Delivery
getPackagesByDroneId(droneId int) : List<Delivery>
```

PathFinder

```
getPath(Position truckPos, Position packagePos) : FlightPlan
```

DroneStarter

`POST <drone_ip>/drone-api/delivery/start`

`JSON Body : FlightPlan`

Le DroneStarter récupère la socket du drone qu'il veut démarrer grâce à l'interface DroneFinder et lui envoie la requête POST ci-dessus.

DeliveryStateNotifier

`POST /truck-api/delivery/`

`JSON Body : { droneId, DeliveryState }`

PositionAccessor

`GET <drone_ip>/drone-api/position/`

DroneStateNotifier

`droneDown(droneId long) : void`

DroneWatcher

`track(droneId long) : void`

`untrack(droneId long) : void`

PackageModifier

`setPackageStatus(packageId long, PackageStatus status)`

DroneModifier

`setDroneStatus(droneId int, DroneStatus status)`

`assignDeliveryToDrone(Drone drone, Delivery delivery)`

DroneFinder

`getAvailableDrones() : List<Drone>`

`findDroneById(droneId long) : Drone`

WarehouseAPI (initialise la connexion avec la warehouse)

`useWarehouseAPI(WarehouseAPI warehouse)`

`sendNotification()`

DroneRegistration (ajouté pour faciliter les tests)

`registerDrone(Drone drone)`

PackageRegistration (ajouté pour faciliter les tests)

`registerDelivery(Delivery d)`

NotificationModifier

```
deleteNotificationsByIds(List<Long> notificationsIds)
```

NotificationRegistration

```
createNotification(long packageId, int deliverySate)
registerNotification(Notification n)
```

NotificationFinder

```
getAllNotification()
```

PackageSelector

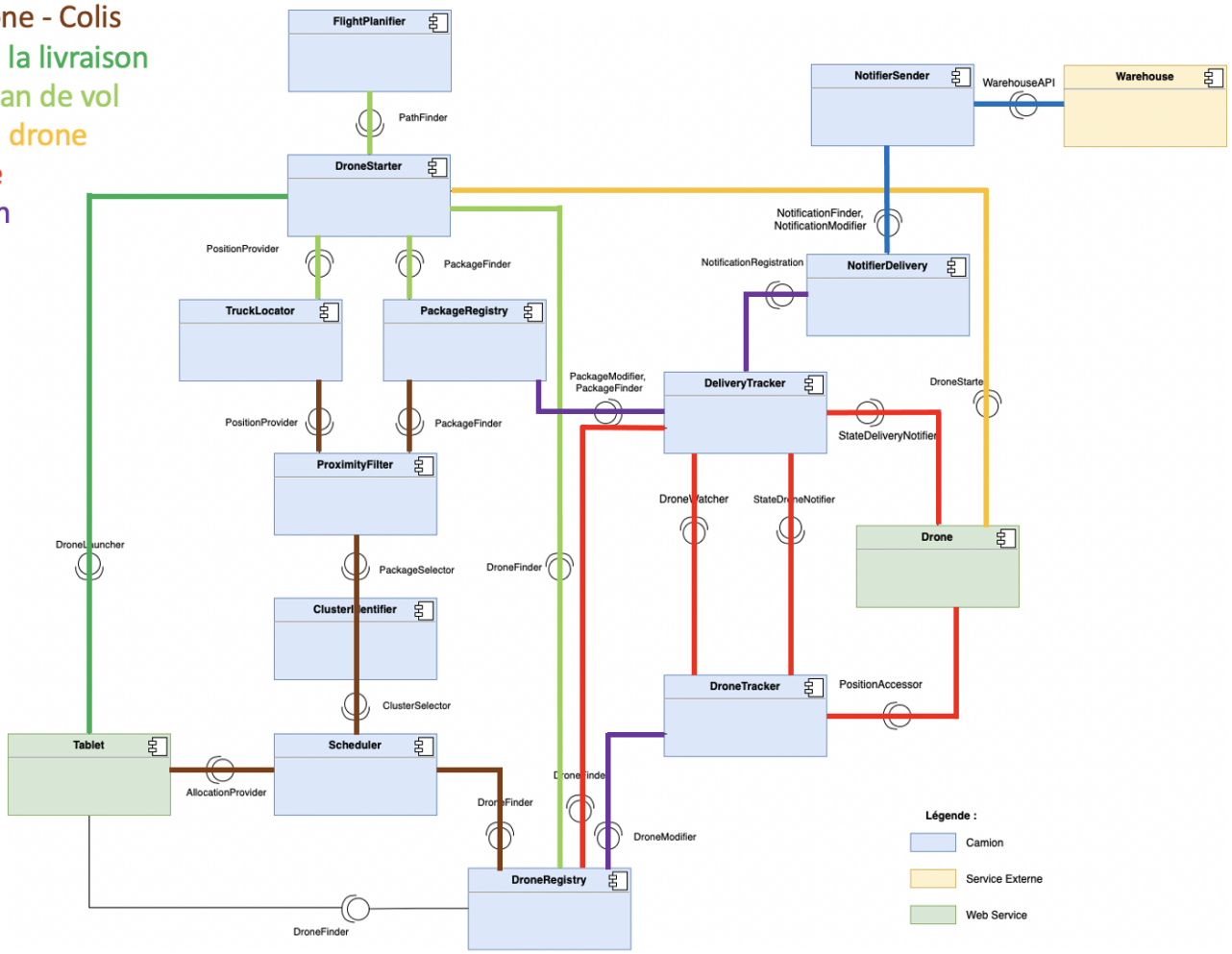
```
getDeliverySelected() : List<Delivery>
```

ClusterSelector

```
getDeliveryCluster(int capacity) : List<Delivery>
```

Déroulement du scénario 1 dans notre architecture

1) Allocation Drone - Colis
2) Lancement de la livraison
3) Création du plan de vol
4) Lancement du drone
5) Suivi du drone
6) Fin de livraison
7) Notification



Le conducteur arrive sur le lieu de déploiement des drones et démarre sa tablette.

La Tablet envoie un signal de départ au composant Scheduler, celui-ci récupère les colis livrables via ProximityFilter et les assigne à un drone récupéré depuis le DroneRegistry. Pour ce faire, ProximityFilter lit la position du camion depuis un capteur via TruckLocator et récupère également les paquets à livrer auprès de PackageRegistry. (Le composant ClusterIdentifier permettant d'optimiser les livraisons en regroupant des colis par zone de livraison n'a pas été implémenté).

Le conducteur lit sur sa tablette les colis qu'il doit associer aux drones libres.

Tablet demande au Scheduler les affectations des drones aux paquets.

Il charge le colis sur le drone et indique que le colis est chargé sur la tablette.

Tablet indique l'identifiant du drone et celui du paquet au DroneStarter.

Le camion lui communique l'itinéraire de livraison.

Celui-ci, après avoir récupéré la position actuelle du camion (via l'interface PositionProvider) et l'adresse de livraison (via l'interface PackageFinder) va alors demander le calcul de l'itinéraire à FlightPlanifier qui le retourne au composant DroneStarter. Après avoir récupéré l'ip du drone (via l'interface DroneFinder) ce dernier envoie l'itinéraire au Drone via l'interface DroneStarter.

Le drone signale qu'il est parti pour sa livraison et commence la navigation.

Drone va envoyer un signal au DeliveryTracker pour que celui-ci mette à jour l'état de la livraison. Par la même occasion, il demande à DroneTracker de commencer à pister le drone concerné. Lorsque DeliveryTracker reçoit une demande de pistage, il met à jour la disponibilité du drone via DroneRegistry.

Le camion surveille la position du drone.

Périodiquement, DroneTracker va envoyer un signal à Drone pour lui demander sa position, il amorce alors un timeout pour considérer ou non un drone comme perdu. Celui-ci lui répond alors avec sa position et DroneTracker met alors à jour le timeout.

Il arrive à l'adresse et dépose le colis, il n'est alors pas connecté au camion.

Drone envoie un signal au DeliveryTracker pour l'avertir de changer son état.

Il repart pour rejoindre le camion.

Arrivé dans la zone de couverture du camion, il signale qu'il a livré le colis.

Drone parvient à envoyer le nouveau statut de livraison au DeliveryTracker qui indique également à NotifierDelivery que le colis livré est arrivé à sa destination.

Le camion signale que le colis est livré dès qu'il peut se connecter à internet

NotifierDelivery effectue une vérification de connexion qui aboutit, il récupère alors les livraisons qui n'ont pas encore été communiquées et les envoie à la Warehouse.

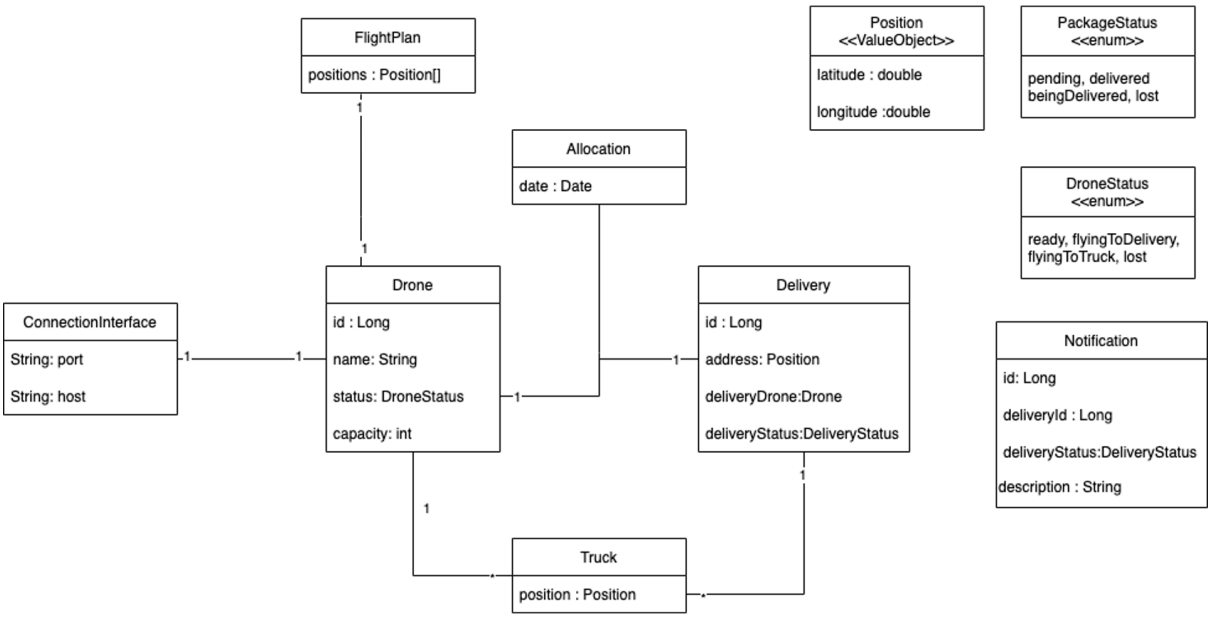
Le drone signale qu'il est retourné sur le camion et qu'il est libre

Drone parvient à envoyer le nouveau statut de livraison au DeliveryTracker qui demande à DroneTracker d'arrêter de pister le drone. DeliveryTracker indique également à NotifierDelivery que le colis livré est arrivé à sa destination. Lorsque DeliveryTracker reçoit une demande d'arrêt de pistage, il met à jour la disponibilité du drone via DroneRegistry.

Le système recalcule l'allocation des colis pour savoir quel colis il lui donne

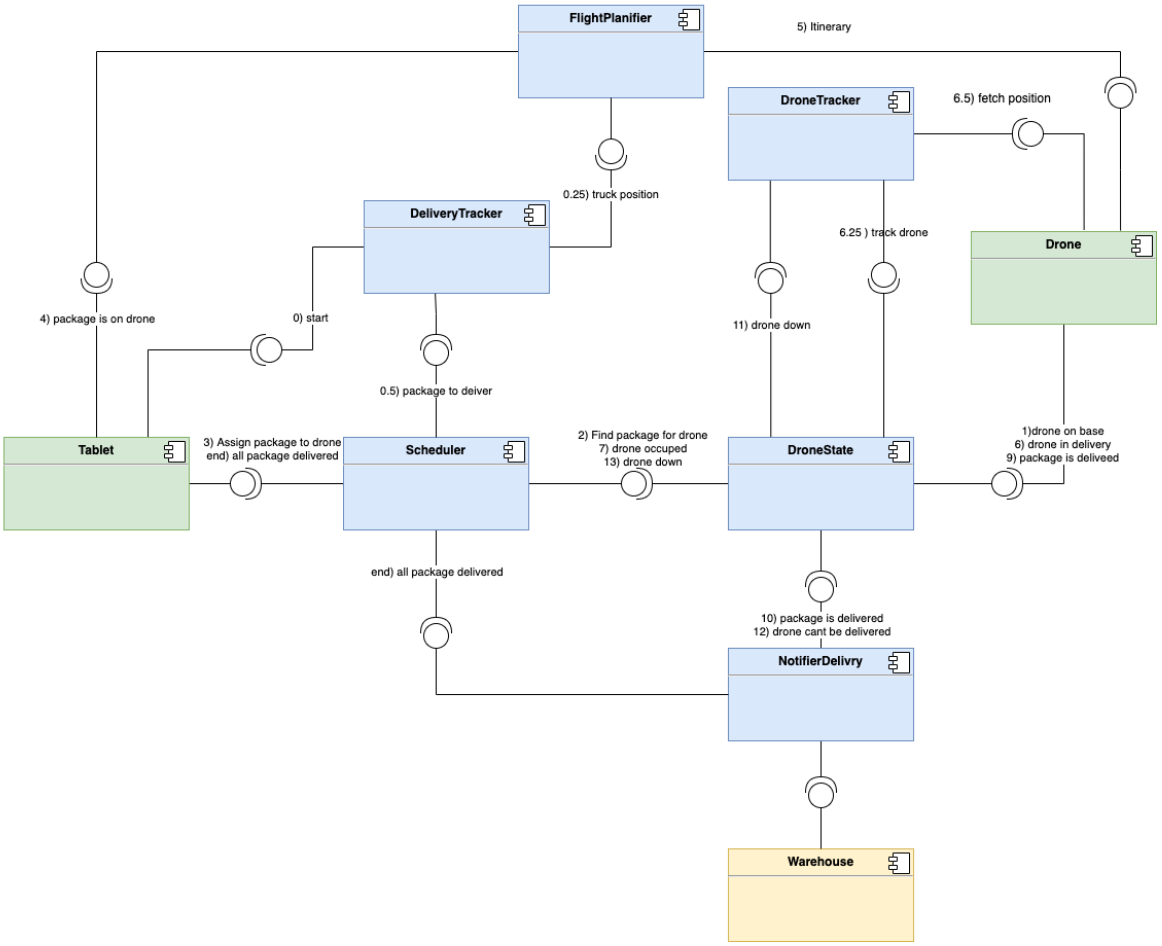
Tablet peut lancer le drone sur son affectation suivante.

Modèle de données



Evolution de l'architecture

Première Version de notre architecture :



c. ***Architecture technique***

Nous nous sommes orientés vers du java pour l'application du serveur embarqué dans le camion.
Plus précisément , nous avons choisi le framework Spring-boot comme principale technologie pour les raisons suivantes :

- Adapté pour faire du N-Tier
- Tâche CRON pour le suivi des drones en livraison
- Son utilisation demande d'importantes ressources (notamment en termes de mémoire), cependant notre camion à la capacité d'embarquer le matériel nécessaire.
- Limite : le camion peut être down mais pas de contrainte d'être opérationnel 24/24
- Gestion facile de la persistance avec openJPA.

La tablette du conducteur, qui sert à connaître les assignations de paquets aux drones et à démarrer les drones, fut mockée en python. Il s'agit d'une simple CLI, ainsi nous avons opté pour une technologie proposant des exemples pour réaliser rapidement et simplement des CLIs.

Quant aux mocks des drones et de la warehouse, ceux-ci ont été réalisés en node js.
Les raisons pour ce choix de technologies sont les suivantes :

- Facilité de développement
- Rapidité de mise en place d'un serveur et d'une api avec ExpressJS (En 2 min un serveur est up)
- Point de connexion Wifi: les drones communiquent en Wifi avec le camion qui présente un point de connexion. Le camion aurait une antenne Wifi qui pourrait permettre de livrer 30 km au environ
- Pour une évolution future, cette technologie permet aussi l'envoi de vidéos.
- Enfin, il existe déjà des versions embarquées de node js s'interfaçant avec du langage C++ (pour l'embarcation du code sur les drones). La documentation de node js embedded est disponible ici : https://nodejs.org/api/embedding.html

4. ***Rétrospective***

Notre application est stable et fonctionnelle et pourrait être, demain, déployée dans un camion avec de vrais drones. Cependant nous avons identifié quelques faiblesses dans notre architecture.

Les communications camion-drone sont effectuées via WIFI avec un protocole HTTP REST (fonctionne et reste plausible) mais des protocoles plus légers existent et seraient mieux adaptés à la campagne où les déconnexions sont fréquentes (protocole SIP (Session Initiation Protocol) ?).

De plus, notre application n'intègre actuellement aucune mesure de sécurité : une personne lambda pourrait très bien se connecter au drone et lui envoyer un nouvel itinéraire. Le framework *Spring Security* pourrait être utilisé pour y remédier.

Notre architecture n'intègre actuellement aucune vérification des informations qui sont envoyées par la tablette : le livreur peut choisir les actions à effectuer dessus.

Nous envisageons d'implémenter de nouvelles fonctionnalités pour optimiser nos livraisons. Par exemple, le contournement d'obstacle grâce à une reconnexion au camion : si le drone rencontre un obstacle sur son trajet, il tente de faire un léger contournement (quelques mètres) si ce n'est pas possible il va se reconnecter au camion pour demander un itinéraire alternatif.

Nos drones doivent également prendre en considération les facteurs météorologiques. Le camion pourrait récupérer des données météorologiques ou un drone pourrait même le prévenir d'une zone affectée par des rafales de vents. Ainsi le camion adaptera les prochains itinéraires.

Nous souhaitons également mettre en place un meilleur suivi de nos drones. Lorsque le camion n'arrive plus à communiquer avec un drone, ce dernier est rapidement considéré comme perdu. Nous pouvons envisager qu'un second drone, volant dans le même secteur, serve de relais au camion et tente de communiquer avec le premier drone. Cela nous permettrait de limiter les pertes des drones.

5. ***Auto Évaluation***

David	Kevin	Martin	Nathan	Thibaut
100	100	100	100	100

6. ***Bibliographie***

- Spring-Boot : https://spring.io/projects/spring-boot
- Node JS : https://nodejs.org/en/about/
- Spring Security : https://spring.io/projects/spring-security
- Protocole SIP : https://en.wikipedia.org/wiki/Session_Initiation_Protocol
- Jean-Aimé Maxa. Architecture de communication sécurisée d'une flotte de drones. Réseaux et télécommunications [cs.NI]. Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier), 2017. Français. ffNNT : 2017TOU30102ff. fftel-01570242v2 https://tel.archives-ouvertes.fr/tel-01570242v2/document