Rapport Architecture ISA-DevOps

Drone Delivery : Livrair

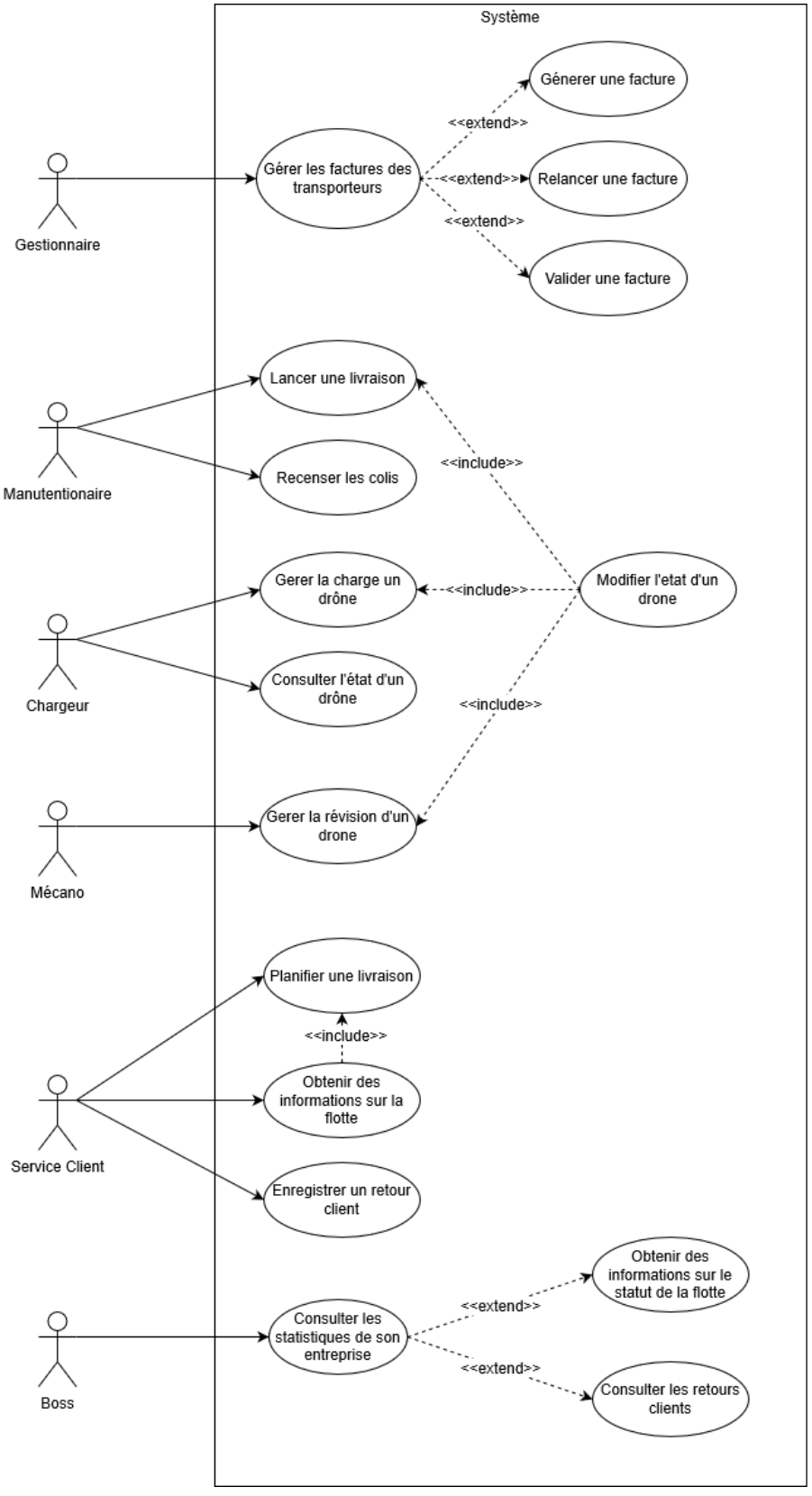
Université Nice Sophia Antipolis | POLYTECH NICE-SOPHIA

Membre de UNIVERSITÉ CÔTE D'AZUR

Pierre-Antoine Boulat - Sylvain Masia - Richard Peres - Maxime Leras - Anthony Choquard

Sommaire

Use Case


Système
Gestionnaire
Gérer les factures des transporteurs
Générer une facture
Relancer une facture
Valider une facture
<<extend>>
<<extend>>
<<extend>>
Manutentionaire
Lancer une livraison
Recenser les colis
<<include>>
Chargeur
Gerer la charge un drône
Consulter l'état d'un drône
Modifier l'etat d'un drone
<<include>>
<<include>>
Mécano
Gerer la révision d'un drone
Service Client
Planifier une livraison
<<include>>
Obtenir des informations sur la flotte
Enregistrer un retour client
Boss
Consulter les statistiques de son entreprise
Obtenir des informations sur le statut de la flotte
Consulter les retours clients
<<extend>>
<<extend>>

Explication des choix

Concernant **le gestionnaire**, il n'a qu'une seule tâche directe, étant celle de gérer les factures. Cependant cette tache s'étend en trois autres sous-tâches : il peut donc générer une nouvelle facture à la fin de la journée pour chaque transporteur, valider une ancienne facture après que le paiement ait été effectué ou relancer un fournisseur qui n'aurait pas réglé sa facture dans le temps donné.

Le manutentionnaire est responsable du lancement des livraisons. De ce fait, après avoir récupéré les informations sur les différentes livraisons prévus par **le service client**, il associe physiquement le colis à son drone, et une fois placé sur l'aire de lancement, il renseigne au système que la livraison a commencé. D'autre part, lorsque les transporteurs viennent déposer les différents colis au dépot, il se charge de les recenser dans le système.

Concernant **le chargeur**, lors de la réception d'un drone après livraison, il est chargé de vérifier l'état physique du drone. On suppose ici que les capteurs du drone fourniront des données tel que la charge, le temps de vol depuis la dernière révision ou encore l'état de ses composants. De ce fait, le responsable de la charge devra obtenir des informations sur le drone, et modifier son état lorsqu'il choisira de la mettre physiquement en charge. Si le drone a besoin d'une révision, alors il lui donnera physiquement au mécanicien qui changera lui même l'etat. Si le drone est opérationnel, le drone est statué à disponible, et rendu **au manutentionnaire**.

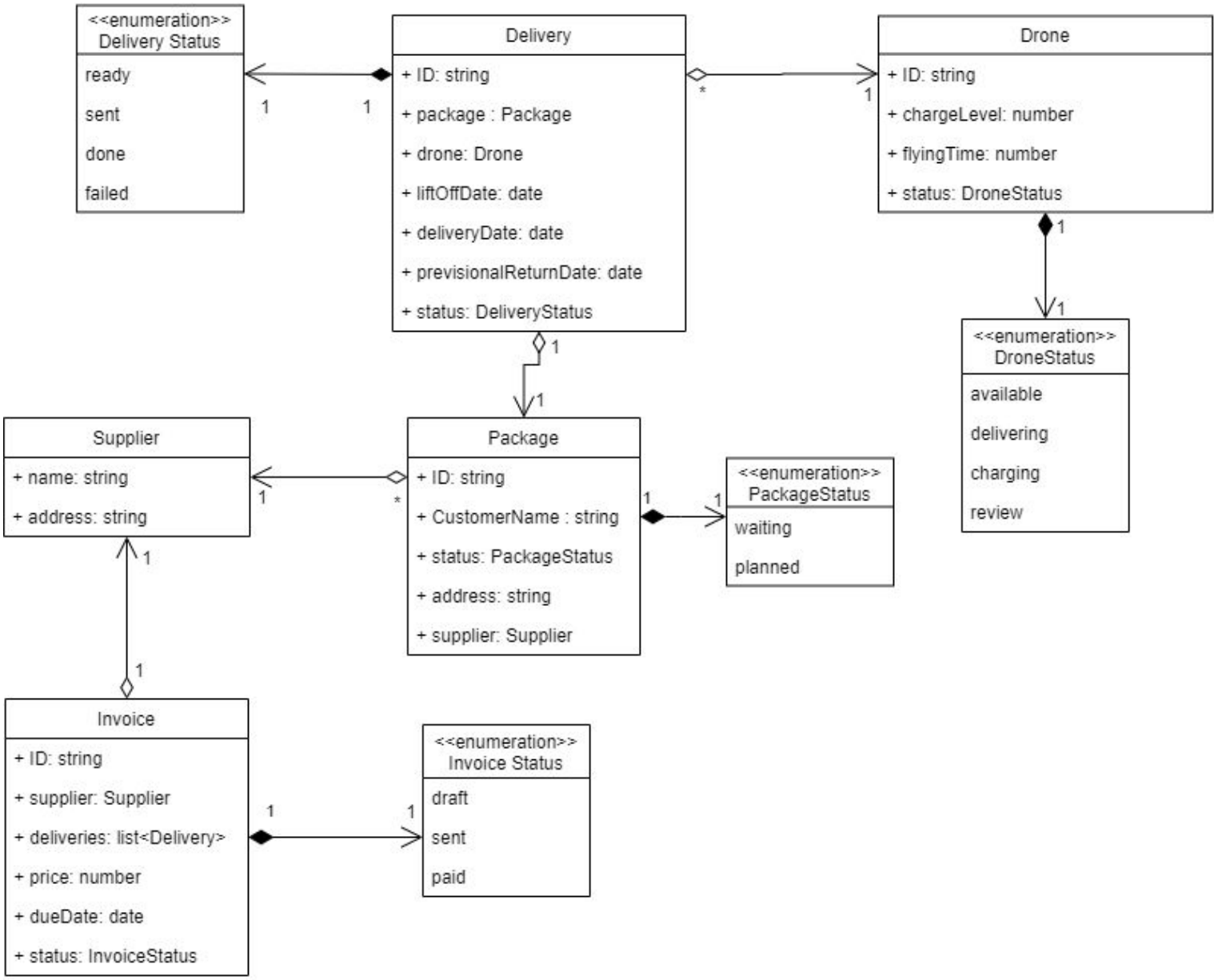
De ce fait, **le mécanicien** pourra si besoin lancer la révision d'un drone, en modifiant son état pour que le système soit informé de son indisponibilité.

Le **manutentionnaire**, le **mécanicien** et le **chargeur**, ont tous une action permettant de changer l'état du drone, qui est représenté dans notre diagramme par une tâches intitulée « modification de l'état du drone » et qui est « include » dans toutes les tâches des acteurs nécessitant de changer un état.

Pour ce qui est de l'acteur en charge du **service client**, c'est lui qui va organiser les livraisons en fonction des disponibilités. Il sera donc responsable de planifier une livraison, à savoir d'associé un colis à un drone. Cependant, pour pouvoir planifier une livraison, il est nécessaire d'obtenir les disponibilité de la flotte. De plus, nous avons choisi de donner, dans un premier temps, la responsabilité au service client d'enregistrer un retour client.

Enfin, le **boss** dispose d'une tâche lui permettant de consulter les statistiques de son entreprise. Pour se faire, il aura accès à l'etat de la flotte des drones, comme le taux d'utilisation des drones, le nombre de drone en réparation, etc ; et également, il pourra obtenir des informations sur la satisfaction client, en se basant sur le taux de livraison réussi, et sur les retours enregistrés par le service client.

Business Object



Explication des choix

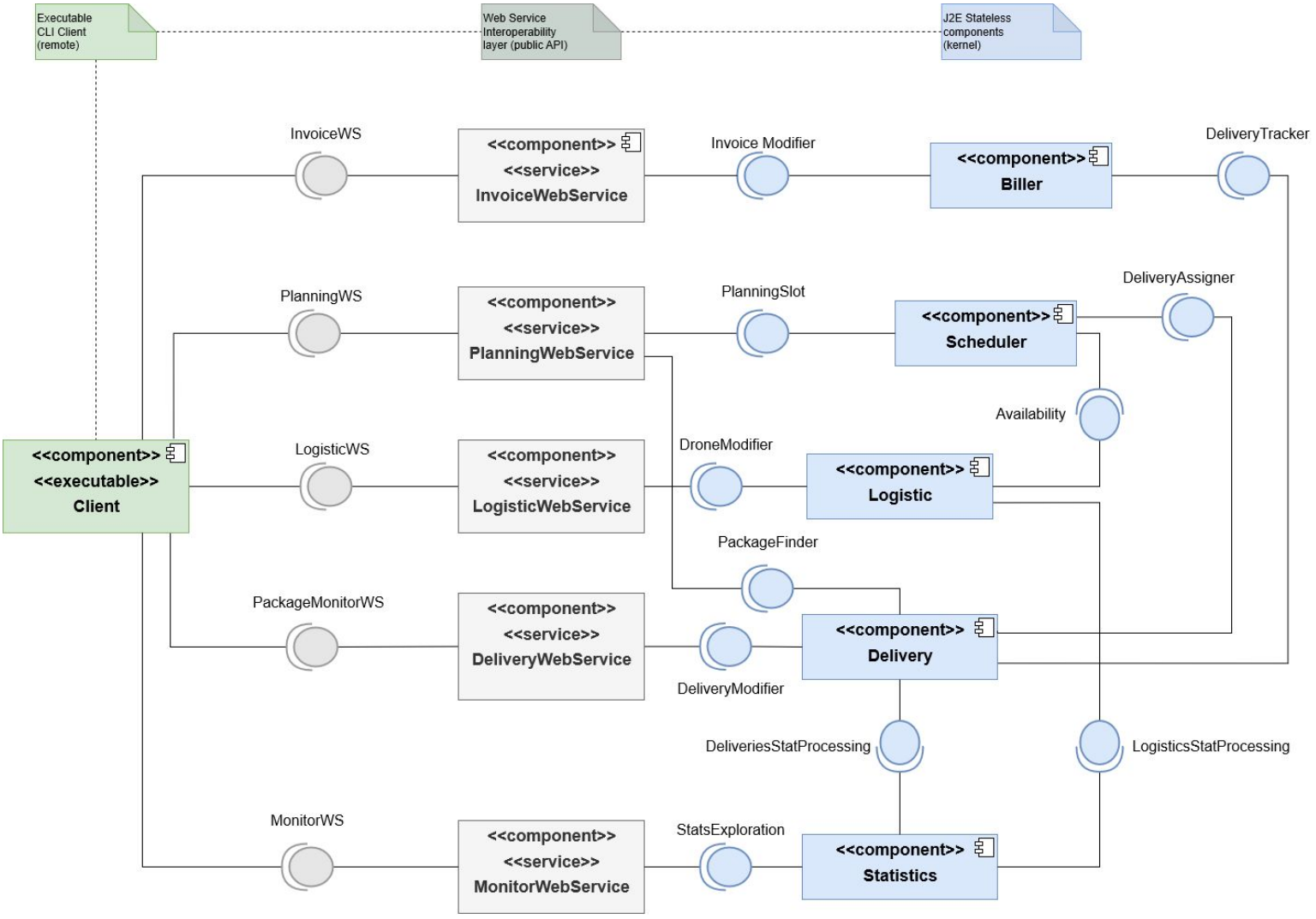
Pour la définition des objets métiers nous pouvons au final couvrir l'ensemble du métier avec peu d'objets.

Les trois objets les plus évidents sont le *Package* qui est l'instance d'un colis, le *Drone,* et le *Supplier* qui représente un fournisseur. Le *Package* contient les informations sur le client et la destination. Il contient également le *Supplier* qui devra être facturé après la livraison. Nous avons fait le choix d'y ajouter un statut afin d'éviter de planifier 2 livraisons pour le même colis par exemple. Le *Drone* quant à lui est responsable de son niveau de charge et de son temps de vol depuis la dernière révision. Il possède lui aussi un statut afin d'aider à la planification des livraisons et d'éviter l'assignement d'un Drone en maintenance à une livraison par exemple.

L'objet qui découle de l'association d'un *Package* avec un *Drone* est la *Delivery,* qui représente la livraison d'un colis par un drone. Une livraison possède une date de décollage, une date de livraison et une date prévisionnelle de retour. Là aussi ces trois attributs permettent la planification de nouvelles livraisons.

Enfin l'objet *Invoice* qui représente la facture envoyée en fin de journée à un *Supplier* est un objet qui sera modifié tout au long de la journée. En effet, dès la première livraison de la journée concernant un fournisseur, une facture est créée et lui est associée. La facture contient donc en fin de journée toutes les livraisons effectuées pour un fournisseur donné. On peut en déduire au moment de l'édition de celle ci le prix total et la date maximale de paiement. Enfin, un statut est donné à la facture afin de par exemple retrouver toutes les factures en retard de paiement.

Component Diagram



Explication des choix

Concernant “l’interface” du client, nous avons choisi dans un premier temps de se limiter à une CLI, qui sera unique pour tous les acteurs. En effet, lorsqu’un acteur lancera la solution, il pourra choisir son “rôle” dans une liste, et obtiendra alors un ensemble de fonctionnalité qui lui seront propre. A terme, nous souhaitons développer une interface graphique qui pourra améliorer la vision de chaque acteur, et faciliter les opérations sur le système.

Nous avons choisi de rendre interopérable via les web services chaque composant indépendamment. Cela se justifie par la faible dépendance des composants entre eux et le fait qu'ils soient tous utilisables par au moins 1 acteur (via le client). Même si comme expliqué précédemment, 1 seul client lourd permet de jouer le rôle de tous les acteurs (et permet donc d'accéder à toutes les fonctionnalités de tous les WS), la multiplicité de ces WS permet dans le futur de faire en sorte que chaque Front spécifique à 1 acteur (et donc à ses besoins) viennent se brancher au(x) WS dont il a besoin. Par exemple, **le boss** pourra avoir un client Front via une appli mobile et lui permettra de monitorer les statistiques en utilisant uniquement le *MonitorWebService*.

Le composant *Logistic* est composé de la partie concernant les drones et leur gestion.

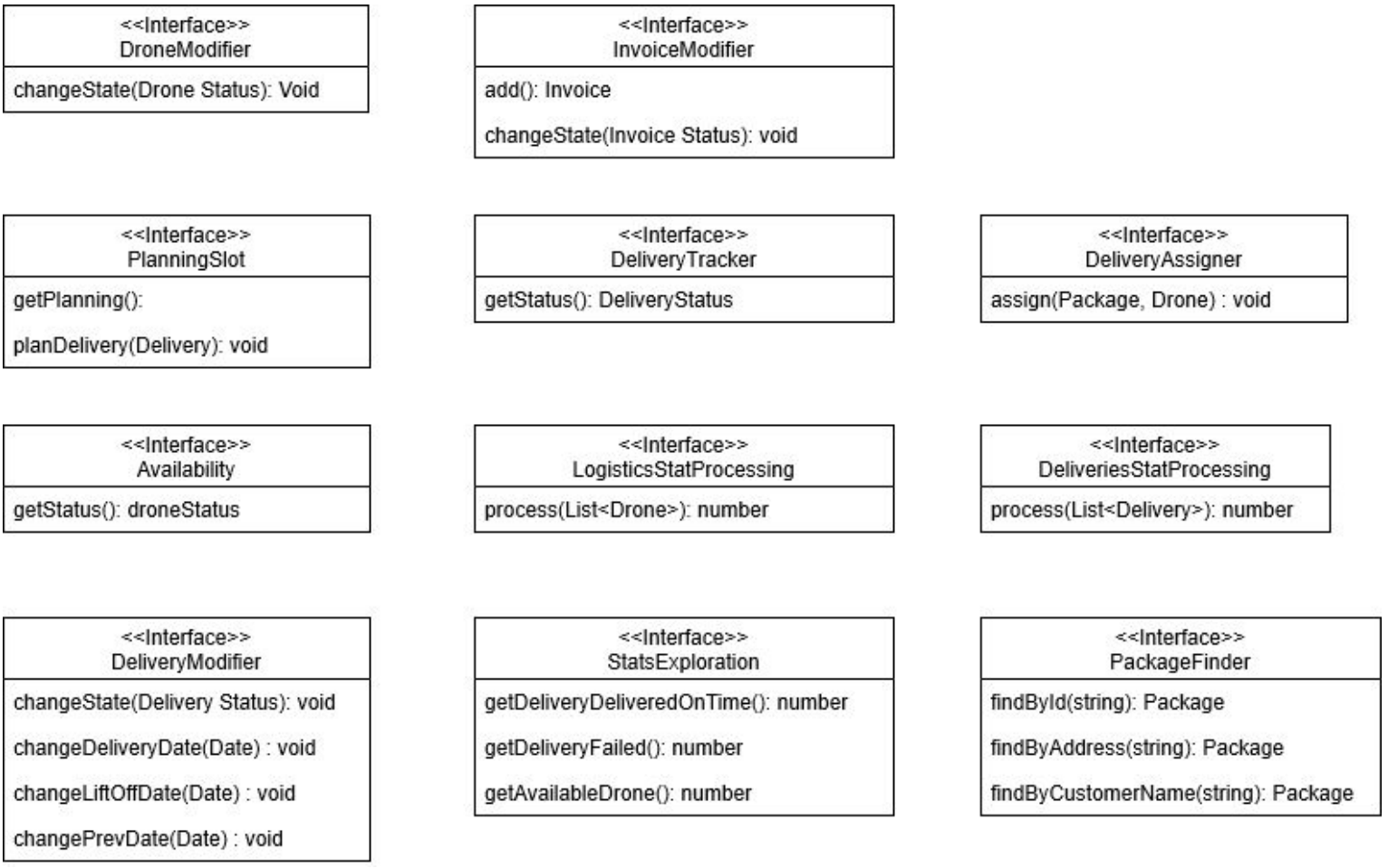
Le composant Delivery regroupe la partie concernant les packages et les livraisons

Le composant *Biller* est chargé de gérer les fournisseurs et les factures. La génération des factures et les relance envoyées aux fournisseurs en retard de paiement.

Le composant *Statistics* permet de consulter et générer les stats d'utilisation des drones et avis clients concernant la livraison.

Le composant *Scheduler* est chargé de planifier les nouvelles livraisons. Pour cela il aura besoin d'une livraison qui lui sera assignée par le component delivery ainsi que d'un drone qui lui sera associé par le component Logistic selon les drones disponible à ce moment.

Interface pseudo-code definition



Explication des choix

PlanningSlot : permet de modifier l'état d'un drone, comme par exemple après le retour d'une livraison si ce dernier est endommagé, ou si il est mis en charge.

InvoiceModifier : permet durant la journée de pouvoir ajouter à une facture une nouvelle livraison d'un paquet relatif à un transporteur. De plus, cette interface permet de modifier l'état d'une facture.

DroneModifier : permet de pouvoir modifier le statut d'un drone pour le rendre accessible, en révision etc...

DeliveryTracker : permet d'accéder à l'état de la livraison.

DeliveryAssigner : permet de fournir une nouvelle livraison au composant scheduler.

Availability : permet d'accéder à l'état d'un drone.

PackageFinder : permet de trouver un colis par son identifiant, le nom du destinataire ou encore l'adresse du destinataire afin de le fournir au composant delivery.

DeliveryModifier : permet de modifier les attributs d'une livraison tels que le statut, la date de livraison, la date de décollage et la date de retour prévue.

StatsExploration : permet d'obtenir les statistiques sur les commandes qui ont été livré avec succès, sur les commandes qui ont échoué, et sur le taux d'occupation des drones.

DeliveriesStatsProcessing : permet de calculer statistiques du nombre de colis livré avec succès ainsi que du nombre de livraison ayant échoué (car le client n'était finalement pas la, par exemple)

LogisticsStatsProcessing : permet de calculer les statistiques sur le nombre de drones utilisés à un instant T.