POLYTECH NICE-SOPHIA

UNIVERSITÉ CÔTE D'AZUR

Projet SI4 - 17 février 2020

Rapport Architecture

Drone Delivery

Groupe N :

Brice Graulier

Théo Fafet

Valentin Campello

Paul Londeix

Armand Boulanger

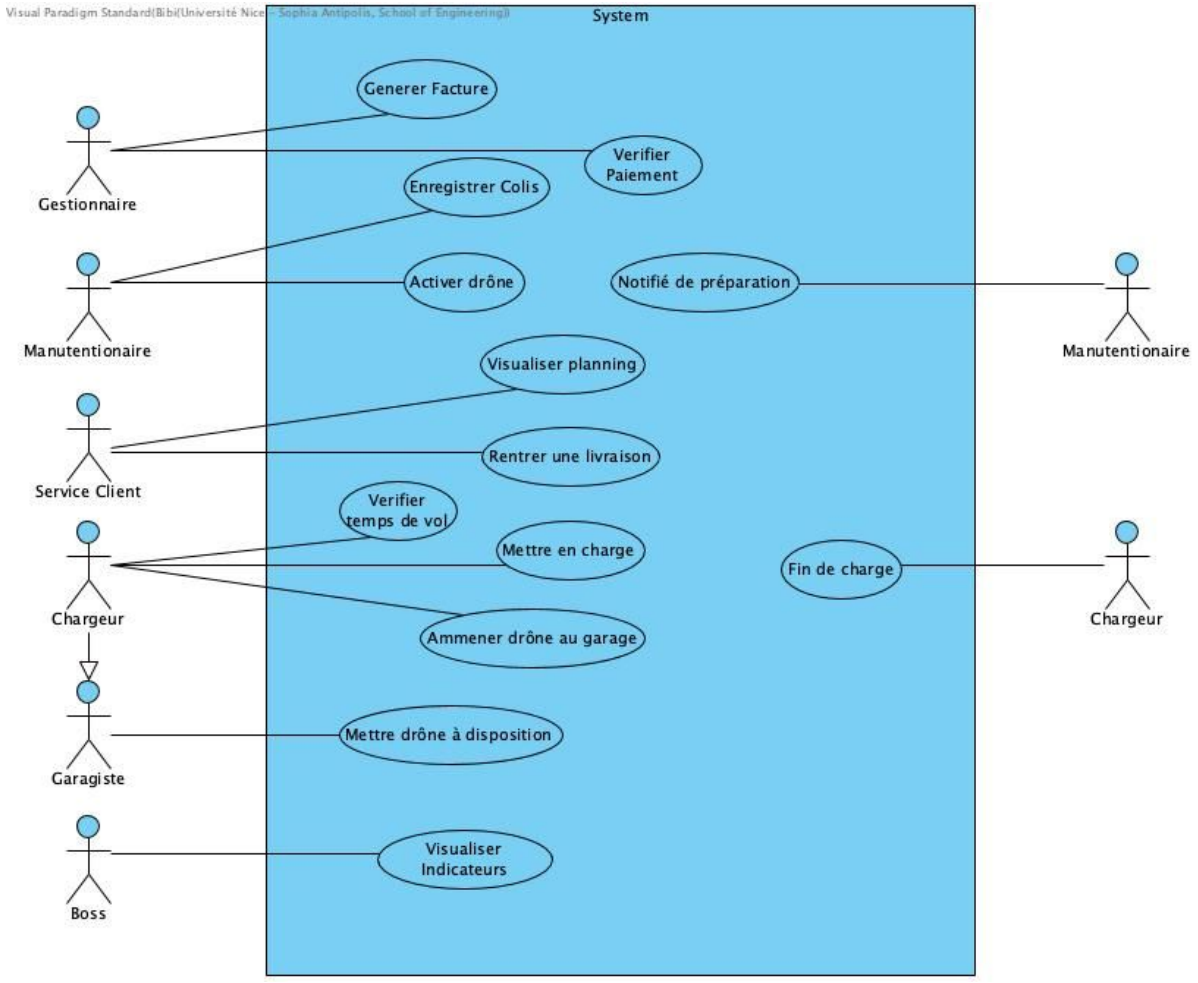
Sommaire

I - Contexte

Livrair est une start-up dont le but est la livraison des colis par drone sur les derniers kilomètres. Ainsi, les transporteurs déposent les colis à Livrair qui s'occupe de les répartir aux domiciles des destinataires. Cette méthode permet aux transporteurs de gagner du temps en évitant les contraintes liées aux déplacements en ville et augmente l'efficacité de la livraison.

La gestion des livraisons et de la flotte de drones nécessite la mise en place d'un logiciel dont l'architecture est décrite dans ce rapport.

II - Cas d'utilisations



Dans ce diagramme des cas d'utilisation, on peut voir que 6 profils différents auront besoin d'accéder au logiciel.

Tout d'abord, le gestionnaire, qui a besoin d'accéder aux factures générées chaque jour, pour pouvoir en vérifier les informations. Par la suite, il voudra y accéder pour vérifier si elles ont été payées.

Le deuxième type d'employé qui aura besoin d'accéder au logiciel est le manutentionnaire. Le manutentionnaire a besoin d'enregistrer les colis dans le système lorsqu'il les reçoit. Ensuite, peu avant une livraison, il nécessite d'être notifié du numéro d'un drone et du colis pour pouvoir les préparer. Enfin, lorsqu'il a fini de préparer le drone et le colis, il a besoin de pouvoir notifier le système que le drone est prêt, pour que celui-ci puisse partir.

Le troisième type d'employé est le service client, le service client reçoit des demandes de la part des clients, par voie téléphonique ou électronique, contenant un nom de client, un colis, et un horaire. Elle a donc besoin de visualiser le planning des drones disponibles, pour pouvoir par la suite réserver un drone pour une livraison dans celui-ci, au plus proche de l'horaire voulue par le client.

Le quatrième type d'employé est le garagiste, qui aura besoin de pouvoir changer le statut d'un drone à la fin de sa maintenance, pour le rendre disponible dans le système.

Le cinquième type d'employé est le chargeur, le chargeur à besoin d'accéder aux informations des drones, pour pouvoir décider ce qu'il doit faire lorsqu'il les récupère. Comme le garagiste, il a besoin de pouvoir changer le statut des drones, en charge quand le drone récupéré n'a pas assez de charge pour refaire un voyage, en maintenance quand il a dépassé les 20h de vol depuis la dernière, et enfin disponible quand il a fini d'être chargé ou qu'il peut encore faire un voyage. De plus, le chargeur a besoin de changer le statut des colis en livré si le drone est revenu sans, dans le cas contraire, remettre leur statut en stock.

Enfin, le boss a besoin d'accéder au taux d'occupation des drones, qui est le nombre de drones en livraison par rapport à la somme des drones disponible et en livraison ainsi qu'au taux de satisfaction des clients, qui sera calculé grâce au taux de livraisons réussies. Pour la version MVP, nous considérons que toutes les commandes sont livrée, ainsi la satisfaction client sera toujours de 100%, cependant l'ajout d'une livraison non-effectuée dans une prochaine version modifiera celle-ci.

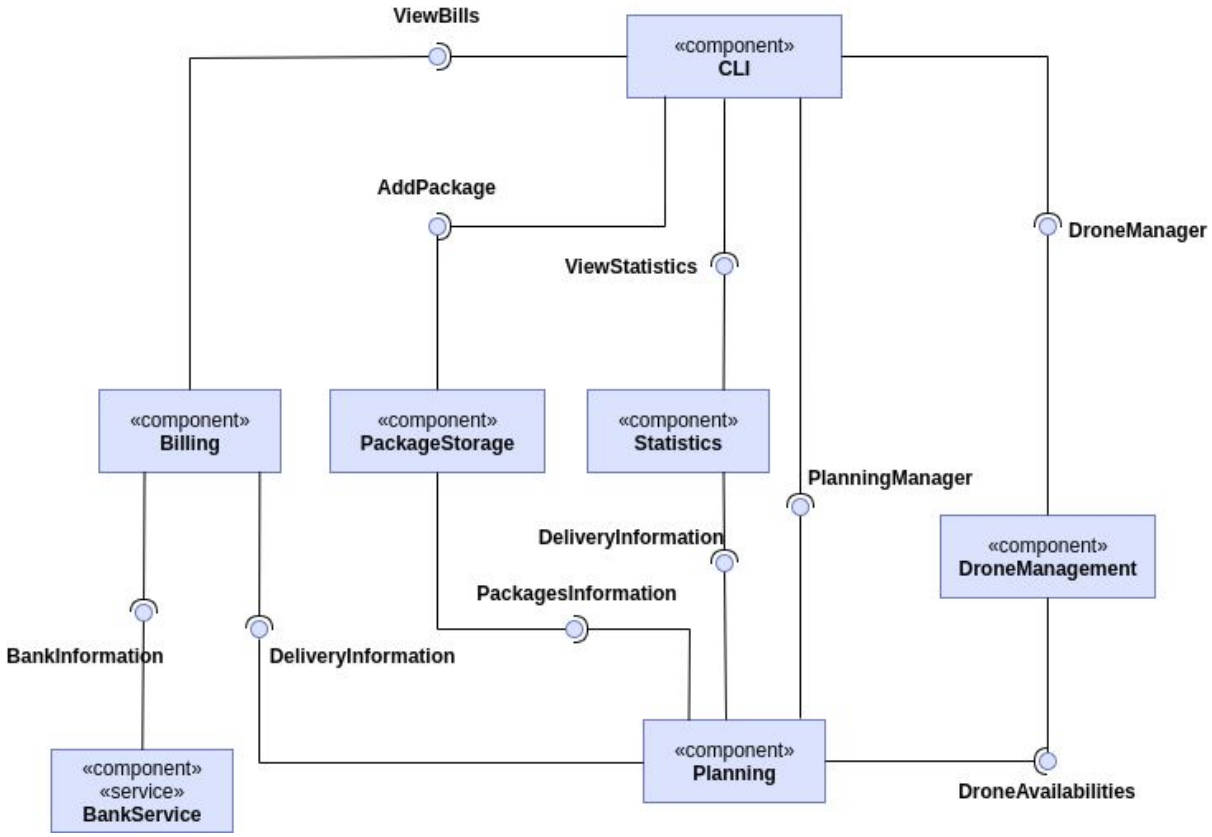
Dans cette première version du logiciel, nous avons fait certains choix. Tout d'abord, nous avons choisi que les colis seraient entrés dans le système par le manutentionnaire lors de la livraison. Il entrera le numéro du colis, le nom et l'adresse du client, le chemin et le temps de livraison seront calculés dans le logiciel et seront mis à disposition du service client lorsqu'il devra planifier la livraison du colis en accord avec l'horaire donnée par le client.

Pour ce qui concerne le gestionnaire, nous avons choisi d'automatiser la création de facture ainsi que la confirmation du paiement, limitant donc son rôle à des vérifications de la génération de facture et du paiement au bout des 30 jours.

Ensuite, sur cette première version, les demandes électroniques des clients seront soumises en dehors du système par mail envoyé au service client.

Enfin, le taux de satisfaction des clients sera le taux de livraisons réussies et non pas une enquête de satisfaction auprès des clients, car, comme pour le point précédent, une interface pour les clients demanderait trop de temps et n'apporterait que peu d'informations pour une première version.

III - Diagramme de composants



IV - Définition des interfaces

DeliveryInformation prototype (Planning → Billing et Planning → Statistics) :
Cette interface permet de fournir les différentes livraisons aux composants *Billing* et *Statistic*.

List<Delivery> getDeliveries(Day);

AddPackage prototype (Package Storage → CLI) :
Cette interface va donner l'option à l'utilisateur (notamment le manutentionnaire) d'entrer les nouveaux paquets livrés dans le système pour qu'ils soient enregistrés et référencés.

void registerPackage(String adresse, String firstName, String lastName, String shipper);

ViewBills prototype (Billing → CLI) :
Cette interface permet de récupérer les factures enregistrées.

void getBillings();

ViewStatistics prototype (Statistics → CLI) :
Cette interface permet de récupérer les différentes statistique demandées par le boss.

```
void getSatisfactionStats();
void getUsageStats();
```

PlanningManager prototype (Planning → CLI) :
Cette interface s'occupe de la gestion générale du planning; elle donne donc la possibilité au service client de visualiser, inscrire et annuler les livraisons. Inscrire une livraison reçoit un booléen pour indiquer si un drone est disponible à cet horaire géré automatiquement par le système.

```
boolean addDelivery(Date, Package);
void cancelDelivery(Delivery);
List<Delivery> getDeliveries();
```

DroneManager prototype (DroneManagement → CLI) :
Cette interface va permettre la gestion de drones, d'abord en offrant le visuel sur la charge et le temps de vol d'un drone, et donc de signaler si une recharge ou maintenance est nécessaire. Elle sera également utilisée pour modifier l'état d'un drone (le passage de disponible à la charge ou maintenance et vis versa).

```
void viewDroneInformation(Drone);
void setToCharge(Drone);
void setToMaintenance(Drone);
void setToAvailable(Drone);
void setToFlying(Drone);
```

DroneAvailabilities prototype (Drone Management → Planning) :
Cette interface donne la liste des drones disponible au *Planning*.

```
List<Drone> getAvailableDrones();
```

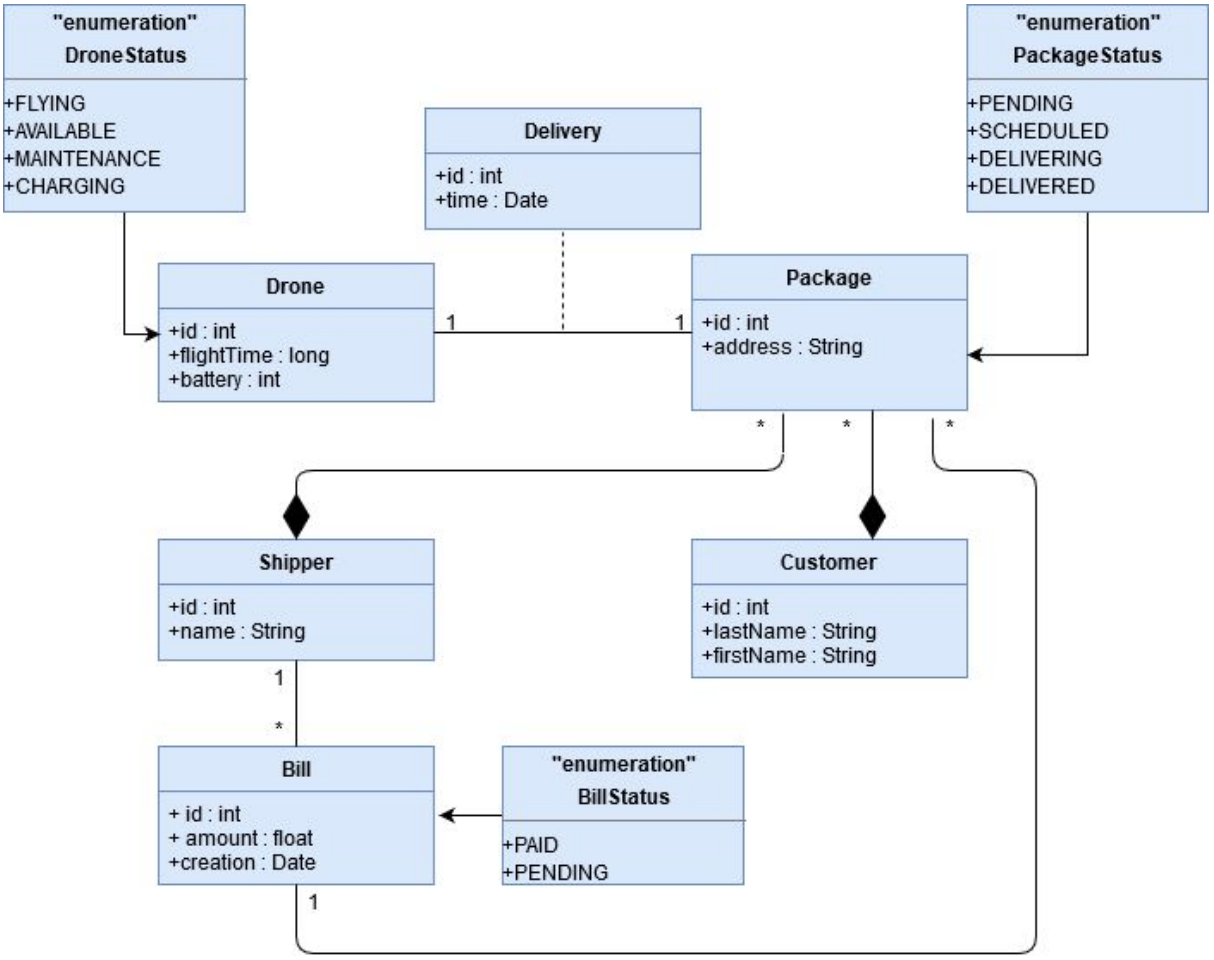
BankInformation prototype (BankService → Billing) :
Cette interface permet à *Billing* d'accéder aux informations du paiement des factures.

```
String billInfo(Bill);
```

PackagesInformation prototype (PackageStorage → Planning) :
Cette interface permet au planning d'accéder aux colis stockés.

List<Package> getPackages();

V - Diagramme de classe des objets métiers



Pour le diagramme de classe des objets métiers du MVP de notre logiciel, nous sommes arrivés au résultat ci-dessus.

La facture doit pouvoir être reliée aux colis correspondant. Nous avons donc fait le choix d'associer la facture à un colis plutôt qu'à une livraison car les informations supplémentaire de la livraison, comme par exemple le drone qui a effectué la livraison, n'ont pas d'intérêts pour la facture.