Isa-Devops - Février 2020

DRONE DELIVERY

Équipe J

Florian AINADOU

Maël DELABY

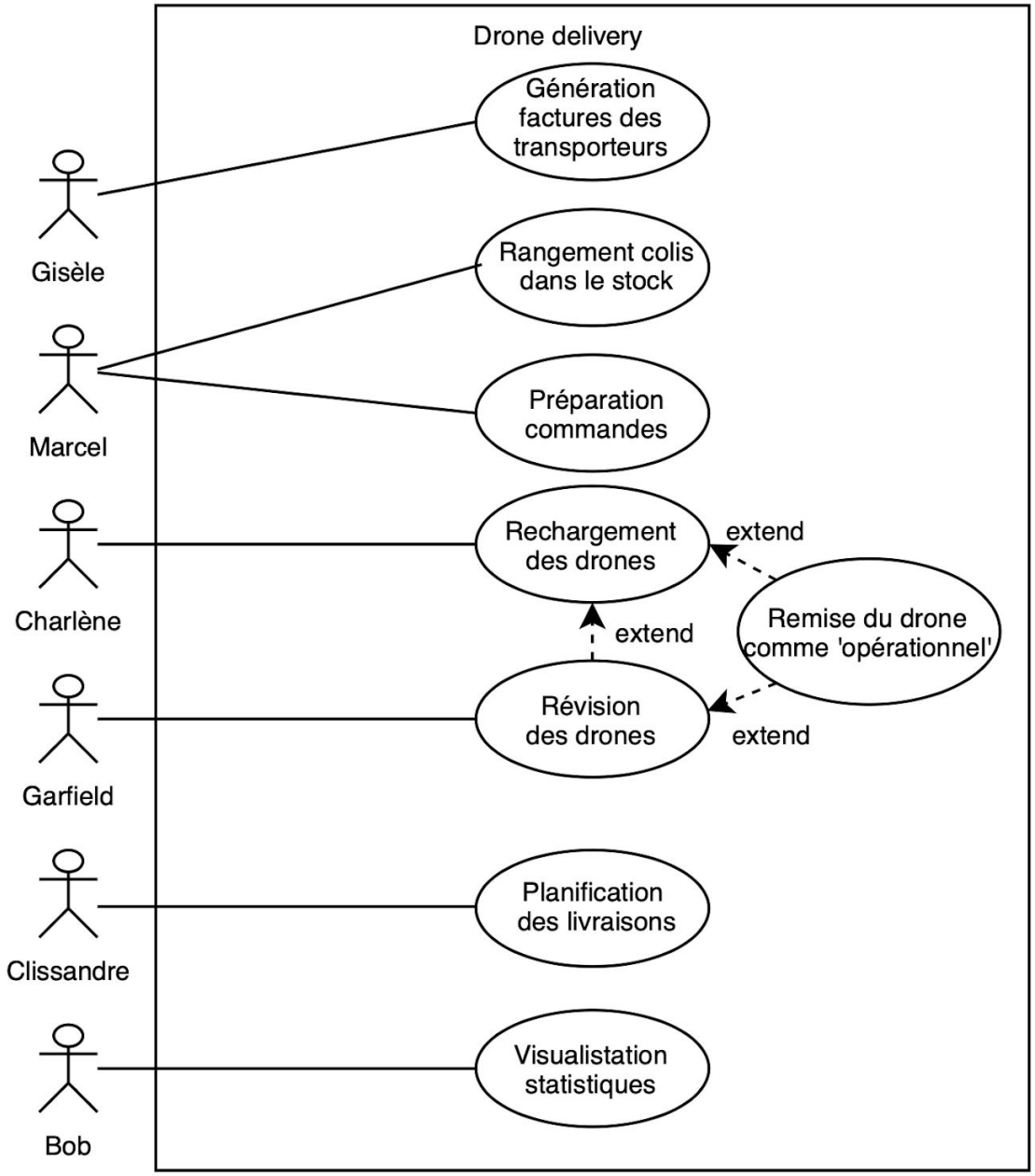
Yannis FALCO

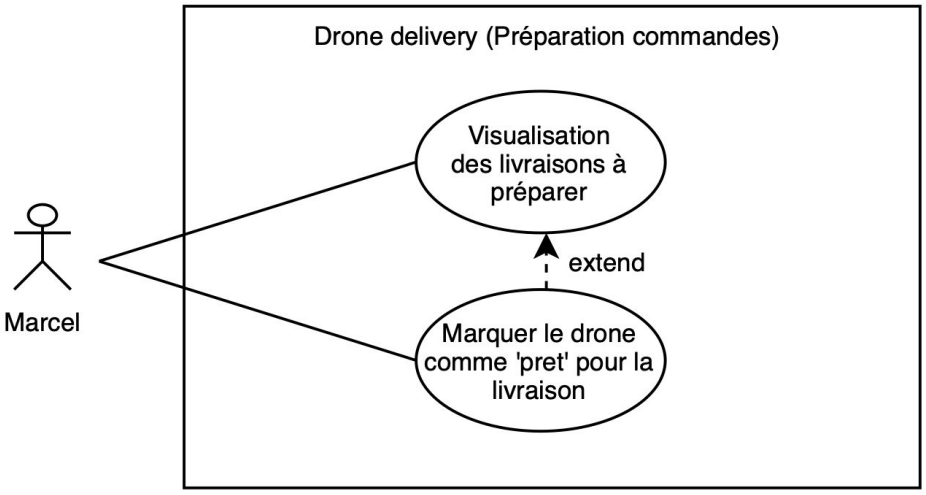
Virgile FANTAUZZI

Rémy LARROYE

Table des matières

Diagramme de cas d'utilisation



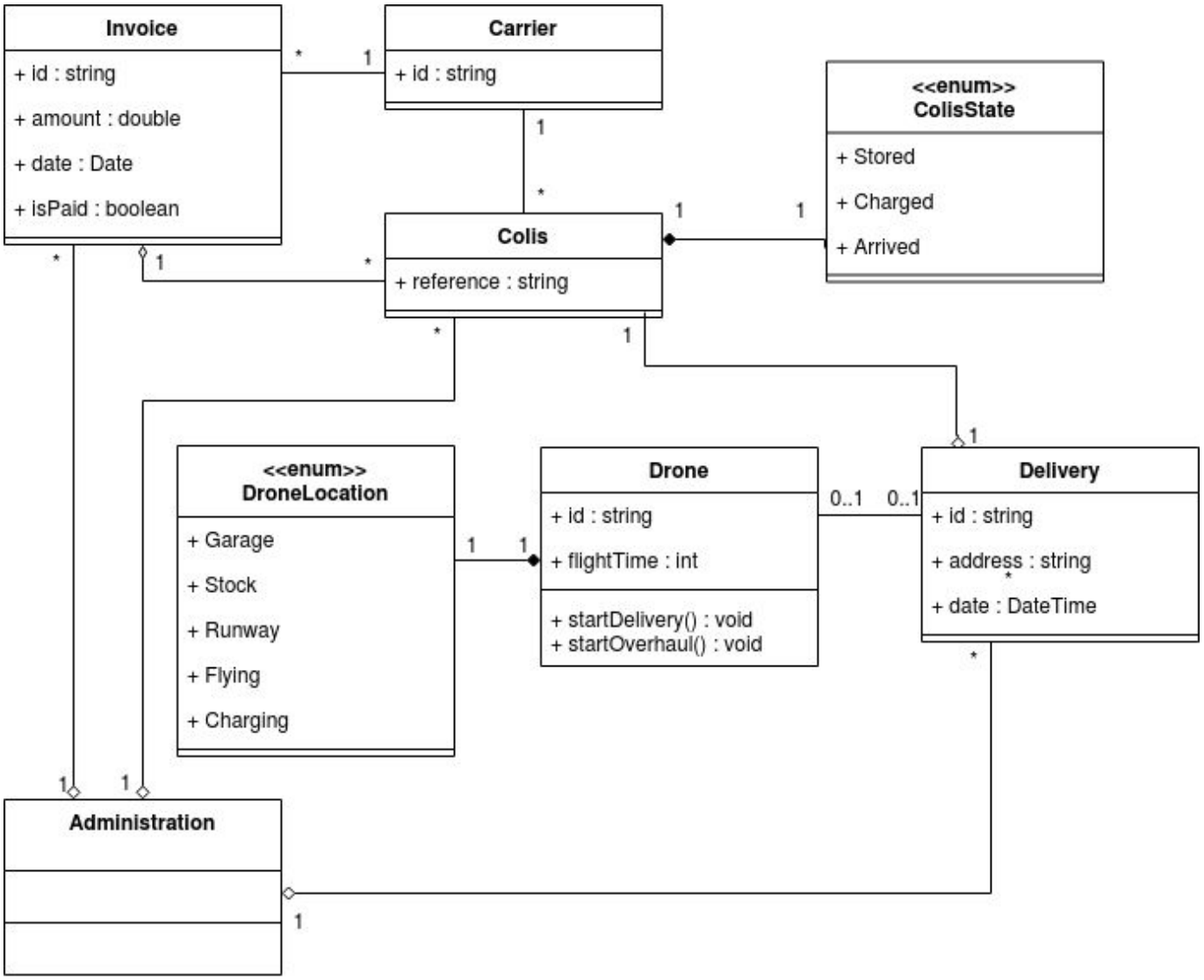


Les éléments interagissants indirectement avec le système n'apparaissent pas sur les diagrammes de cas d'utilisation mais apparaissent dans le diagrammes de composants. Comme la banque, Gisèle communique avec la banque mais ceci est fait en dehors de notre application, et comme les clients avec qui interagit Clissandre.

Les drones n'apparaissent pas non plus, mais il faut qu'il y ai l'envoie du plan de vol (mais cela est géré par notre application).

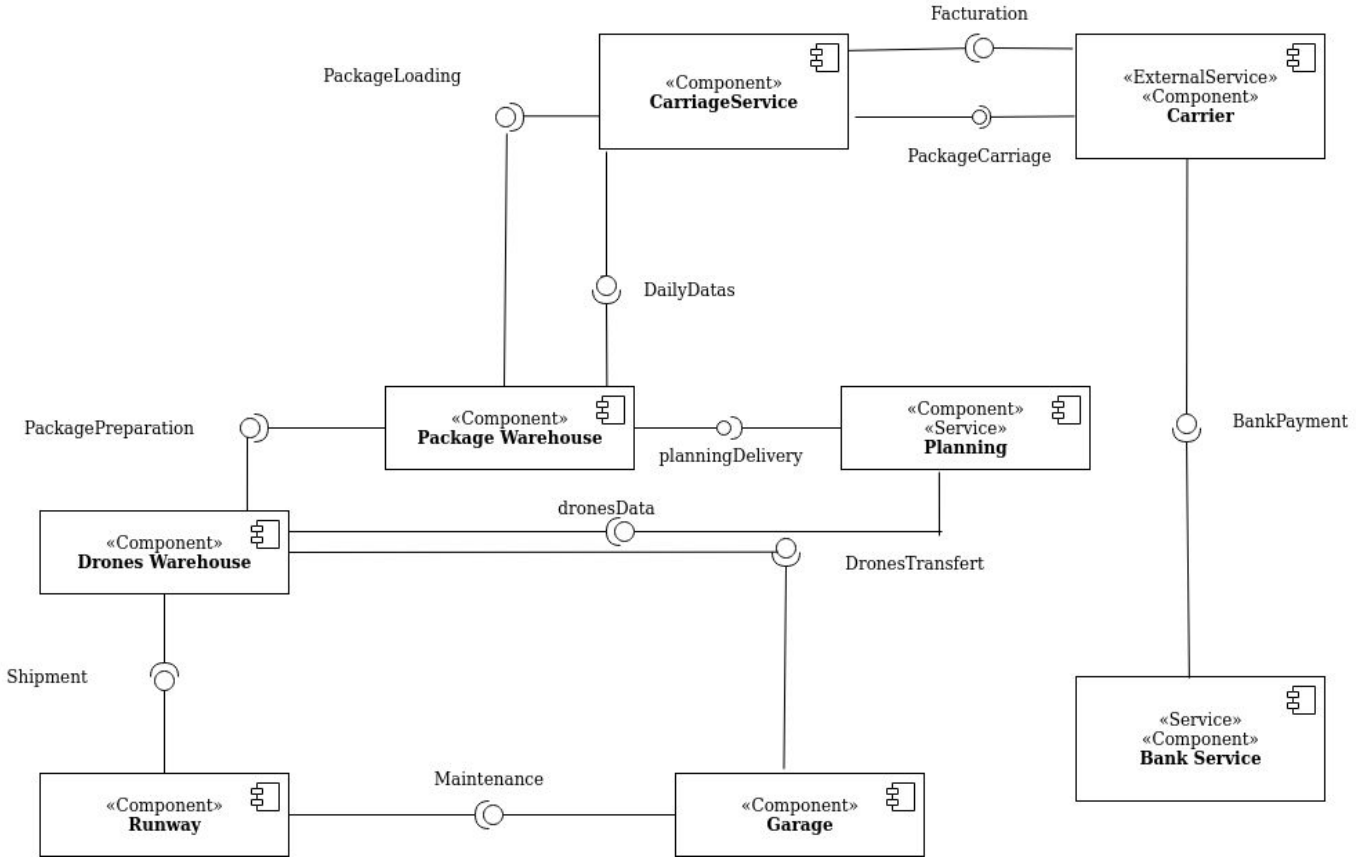
Quand un drone revient de vol Charlène à deux possibilités après avoir rechargé le drone, soit de le remettre dans l'entrepôt pour ses prochaines missions soit de l'emporter à Garfield pour un révision. Cela est représenté par les "extend" des diagrammes.

Diagramme de classes



- Nous avons décidé que le drone avait 0 ou 1 livraison car il est assigné à une livraison lorsque celui ci doit être emporté sur la baie de décollage pour décoller, le reste du temps nous ne lui assignons aucune livraison.
- Nous avons choisi que la livraison avait 0 ou 1 drone car comme pour l'explication précédente, lorsque la livraison est créée, elle peut être pour dans longtemps et donc il faudra assigner le drone à ce moment là et pas de suite
- Nous avons décidé d'implémenter le DP State pour un colis car il suit une ligne dès lors qu'il arrive à l'entrepôt, jusqu'à ce qu'il arrive à destination, ce n'est pas le cas pour le drone qui peut passer de BaieDeDecollage à EnRecharge ou de BaieDeDecollage à Stock par exemple.
- Nous avons décidé de stocker les livraisons, les colis et les factures dans la classe administration afin de faciliter la création de statistiques ainsi que la gestion d'autres comportement comme la gestion des factures.

Diagramme de composants



Description des composants

BankService: ce composant est un composant externe s'occupant du paiement des livraisons journalières de colis par le Carrier. Il est directement relié au Carrier car ce dernier devra payer directement dans un délai de 30 jours à la banque.

Carrier: Ce composant représente le transporteur qui livre des colis et par la suite est facturé pour colis livrés

CarriageService: Ce composant s'occupe de l'enregistrement et de l'acheminement des différents colis vers le PackageWarehouse

PackageWarehouse: Ce composant représente l'entrepôt où sont stockés les colis. C'est à cet endroit qu'on choisit les colis qui seront prochainement livrés. Voulant avoir les informations relatives aux colis envoyés, il fallait un lieu où seraient stockés tous les colis enregistrés par le CarriageService. De ce fait, il sera plus simple de lister les colis restants.

Planning: Ce composant représente le centre de planification des différentes livraisons de colis. En effet, les livraisons doivent être organisées en fonction des demandes des clients. Il était donc préférable de représenter un composant qui serait chargé de planifier les différentes livraisons.

DroneWarehouse: Ce composant représente l'entrepôt où sont stockés les différents drones prêts à être utiliser pour un envoi de colis.

RunWay: Ce composant permet de représenter la baie de lancement et c'est à partir de cet endroit qu'on envoie le drone chargé du colis et c'est en ce lieu que revient le drône.

Garage: On considère ici que cet composant est le lieu où l'état des drones est analysé. S'ils sont en bon état, ils sont directement envoyés au DroneWarehouse, sinon ils seront mis en charge ou réparés avant d'être renvoyé

Nous avons décidé de fusionner le lieu de chargement et de maintenance des drones car ces deux opérations représentent la même chose: l'impossibilité d'utilisation d'un drone pendant un certain temps. Nous avons donc décidé d'en faire un composant unique.

Interfaces pseudo-code definition

```
public interface Shipment {

  void sendShipment(Drone drone, Colis shipment);

  boolean landingDrone(Drone drone);

}
```

C'est l'interface qui permet la communication entre le composant RunWay qui gère les drône en cours de livraison et celui de l'entrepôt DroneWarehouse.

On retrouve donc les fonctions :

Envoie d'un drône en livraison avec le drône et son colis.

Atterrissage d'un drone qui renvoie un boolean si l'atterrissage c'est bien passé ou non

```
public interface Maintenance {

  void droneReviewal(Drone drone);

}
```

C'est l'interface qui permet la communication entre le composant RunWay qui gère les drône en cours de livraison et celui des maintenances Garage.

On retrouve donc la fonction :

Envoyer un drône en maintenance.

```
public interface DronesTransfer {

  boolean droneTransferToWarehouse(Drone drone);

}
```

C'est l'interface qui permet la communication entre le composant DroneWarehouse et celui de l'entrepôt Garage.

On retrouve donc la fonction :

Transférer un drône dans l'entrepôt, qui permet de renvoyer un drône qui a terminé sa maintenance directement dans l' entrepôt

```
public interface PackagePreparation {

  void assignShipmentToDrone(Drone drone, Shipment shipment);

}
```

C’est l’interface qui permet la communication entre le composant DroneWarehouse et celui de l’entrepôt PackageWareHouse.

On retrouve donc la fonction :

Assigner un colis à un drone, qui permet de relier les deux objets afin d'effectuer la livraison

```
public interface PackageLoading {

  boolean receiveShipment(Colis shipment);

}
```

C’est l’interface qui permet la communication entre le composant PackageWarehouse et celui de CarriageService.

On retrouve donc la fonction :

Recevoir un colis, avec un boolean qui renseigne si la réception c’est bien passé.

```
public interface PlanningDelivery {

  void assignScheduleToShipment(Colis shipment, Date date);

}
```

C’est l’interface qui permet la communication entre le composant PackageWarehouse et celui de Planning.

On retrouve donc la fonction :

Assigner un horaire à une livraison avec le colis et la date en paramètre.

```
public interface Billing {
  Billing getBill(Colis shipment);
}
```

C’est l’interface qui permet la communication entre le composant CarriageService et celui de Carrier.

On retrouve donc la fonction :

Récupérer la facture d’un colis précis, afin de pouvoir gérer la facturation

```
public interface PackageCarriage {
  boolean receiveShipment(Colis shipment);
}
```

C’est l’interface qui permet la communication entre le composant Carrier et celui de CarriageService.

On retrouve donc la fonction :

Recevoir un colis depuis le transporteur, avec un boolean qui renseigne si la réception c’est bien passé.

```
public interface DailyDatas {
   Colis [] getShipmentsRemaining(Date date);
}
```

C’est l’interface qui permet la communication entre le composant CarriageService et celui de PackageWarehouse.

On retrouve donc la fonction :

Récupérer les colis d’une journée précise afin de pouvoir faire l’inventaire

```
public interface DroneDatas {
  Drone [] getAvailableDrones();
}
```

C’est l’interface qui permet la communication entre le composant DroneWarehouse et celui de Planning.

On retrouve donc la fonction :

Retrouver la liste des drones qui sont actuellement disponibles pour effectuer une livraison.