Drone Delivery

ISA - DEVOPS

Doan Quang Minh
Focas Florian
Montoya Damien
Rigaut François
Todesco Gabin

Année universitaire 2019 – 2020

Table des matières

Introduction du produit

Dans une société ou l'achat en ligne est devenu un acte quotidien, les entreprises se doivent de livrer en temps et en heure les colis aux clients de plus en plus exigeants. Livrair, une start-up française propose une solution de livraison "de dernier kilomètre" afin de respecter au mieux l'horaire choisi par le destinataire. Livrair récupère les colis de grandes chaînes de distribution tel que UPS, Fedex etc… puis les livrerait par drone aux clients qui auront au préalable contacté l'entreprise pour définir l'horaire de livraison.

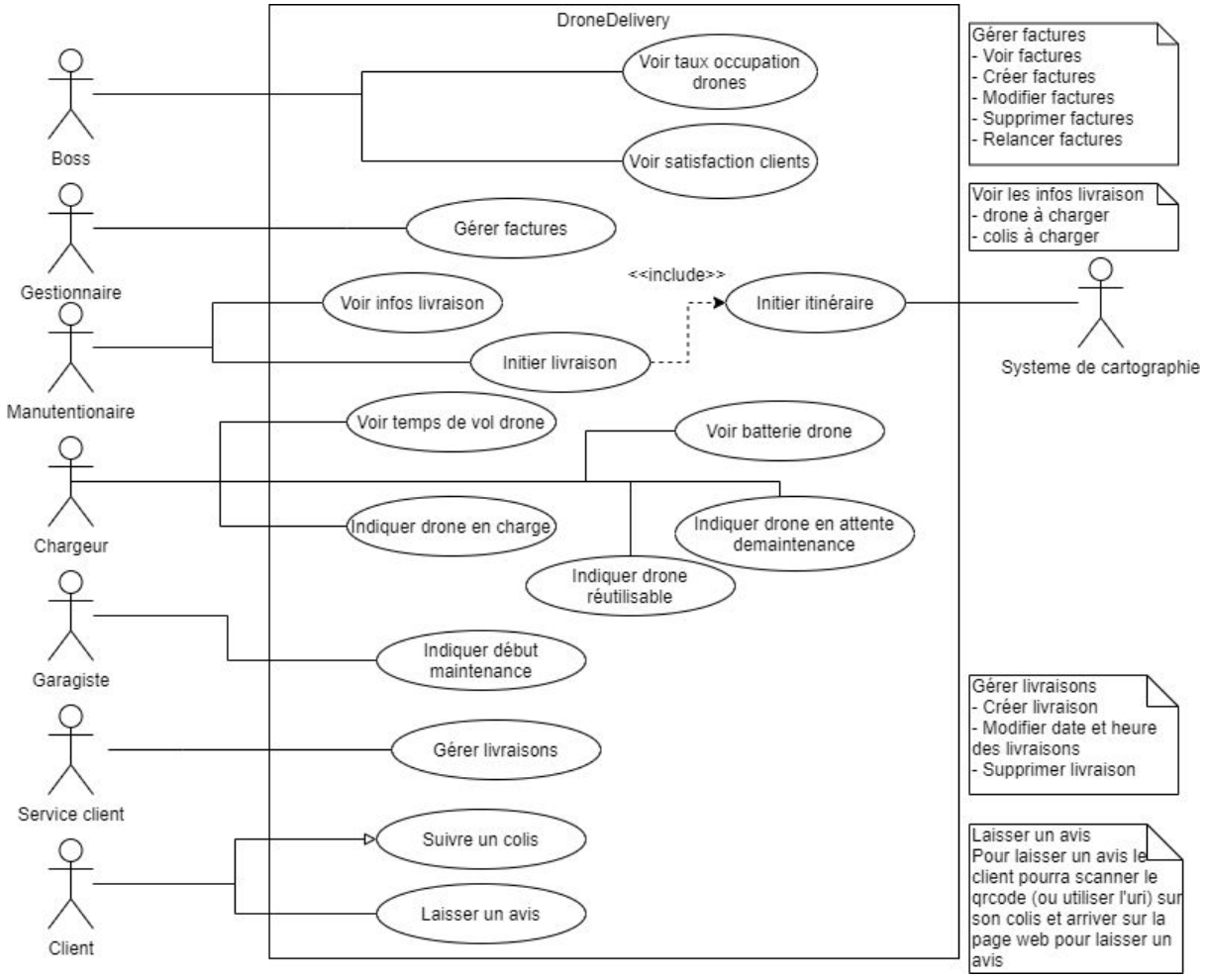
Par souci d'efficacité, l'entreprise gère les colis des clients en les identifiant par leur numéro de suivi de colis chez le fournisseur d'origine. Ces derniers sont alors transmis au client par voie électronique. Cela permet au client d'identifier son colis, et de soumettre des avis sur la qualité de la livraison par voie électronique, car la satisfaction client est un indicateur important pour Livrair.

Deux interfaces sont proposées pour gérer le produit. Une interface logicielle est disponible pour les employés de Livrair pour leur permettre la gestion des drones, des factures etc… Ensuite, une seconde interface est accessible par navigateur et destinée au client pour effectuer le suivi de colis et pour déposer un avis une fois ce dernier reçu. Ces deux parties "front-end" accessible via un navigateur web (ou autre) se connectent à notre partie "back-end".

Grâce aux fonds levés, Livrair dispose de 50 drones pouvant chacun transporter un colis pendant 45 minutes. Après une heure de charge, ce dernier peut repartir effectuer une livraison. Au bout de 20 heures de vol, chaque drone doit passer en révision auprès de notre garagiste interne ce qui prend environ 3 heures.

Diagramme de cas d'utilisation

Suite à notre analyse du produit nous avons pu déterminer les principales actions que nos 7 acteurs pourraient effectuer avec notre système.



Notre analyse nous a permis d'extraire 7 acteurs principaux dont 6 internes à l'entreprise ainsi qu'un acteur externe.

Les 6 acteurs internes à l'entreprise, qui sont le boss, le gestionnaire, le manutentionnaire, le chargeur, le garagiste ainsi que le service client disposent de plusieurs interfaces logicielles pour gérer les différents aspects de la livraison par drone.

Le boss est en mesure de visualiser les statistiques concernant l'entreprise et les clients.

La gestionnaire est en charge de la comptabilité et de la gestion des factures, comprenant la vérification des calculs et de l'envoi quotidien de ces dernières aux fournisseurs. Elle s'assure également du suivi et du paiement des factures.

Le manutentionnaire est la personne devant récupérer les colis provenant des fournisseurs. Il doit également attacher un colis à un drone et de démarrer la séquence de livraison.

Le chargeur doit s'assurer que tous les drône sont chargés. Elle doit également vérifier le nombre d'heures de vol du drone. Si ce dernier a dépassé les 20 heures de trajet, elle doit l'apporter au garagiste.

Le garagiste est, quant à lui, chargé de réparer et effectuer les vérifications mécaniques des drones.

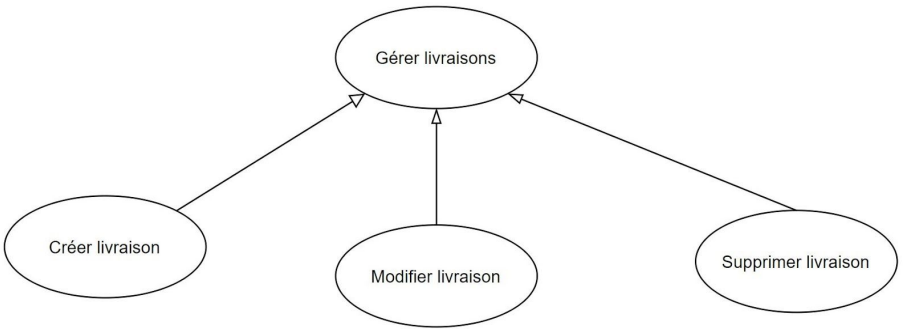
Le service client est en charge de modifier les factures en fonction des demandes de livraisons des clients, que ce soit par téléphone pour par voie électronique. Elle doit donc planifier les livraisons en fonction des horaires et de la disponibilité des drones.

Le client peut utiliser une interface spéciale pour laisser un avis ou suivre un colis. Nous ne mettons pas mis en place de système de compte, mais pour accéder aux informations il doit utiliser le numéro de suivi de colis qui lui est transmis.

Ensuite, Livrair utilise un système de cartographie externe pour définir les itinéraires que les drones doivent emprunter afin d'effectuer la livraison.

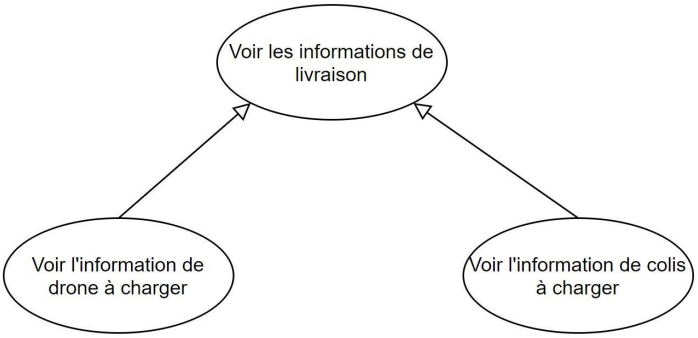
Cas d'utilisation : gestion des livraisons

Notre acteur, Clissandre, responsable de la partie service client de Livrair doit être en mesure d'effectuer diverses opérations concernant les livraisons. C'est elle qui est en charge d'ajouter les livraisons au système un fois les colis et informations reçues. Elle est également en mesure de modifier les informations de livraison comme par exemple la définition de l'horaire de livraison ou même la suppression de celle-ci.



Cas d'utilisation : voir les informations de livraison

L'acteur manutentionnaire est la personne chargée d'attacher les colis aux drones puis d'initier la livraison en chargeant l'itinéraire grâce au service externe de cartographie. De ce fait, le manutentionnaire peut, grâce au système, savoir quel colis charger sur quel drone.



Cas d’utilisation : gestion des factures

Concernant la gestion des factures et de la comptabilité, c’est l’acteur gestionnaire qui s’en occupe. Ce dernier est capable grâce au système de visualiser toutes les factures ainsi que de les manipuler. Le système peut créer toutes les factures du mois. Dans le cas où un fournisseur ne paye pas la facture, une relance peut être effectuée directement depuis le système.

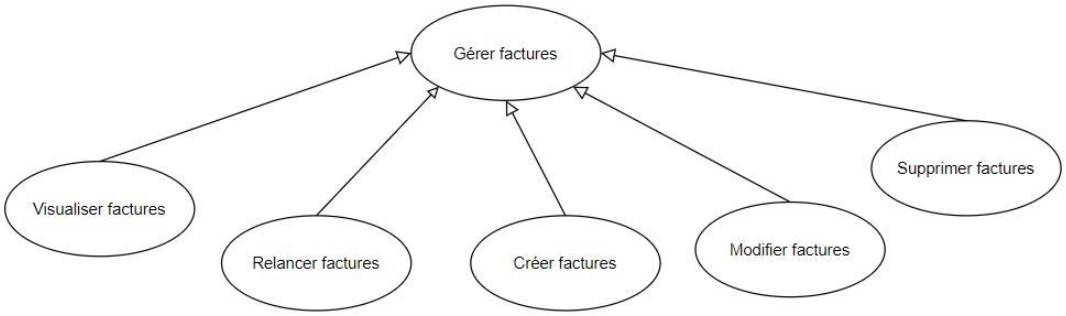


Diagramme de classes

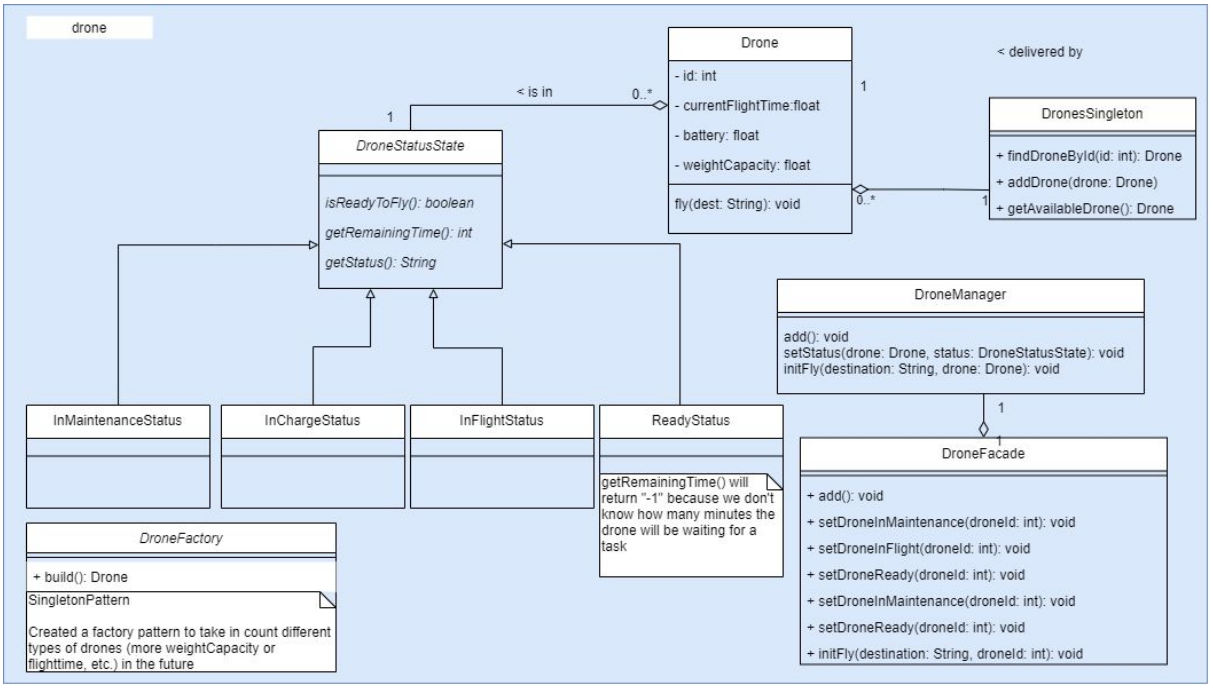
Après avoir défini les différents acteurs de notre système, nous avons pu étudier l'architecture et la structure du projet pour en extraire un diagramme de classe. Nous avons séparé ce dernier en divers groupes pour qu'il soit plus facilement présentable et pour comprendre au mieux sa structure.

Pour mettre en place notre diagramme de classe, nous avons vu l'utilité d'utiliser certains patterns qui seront mis en place plusieurs fois. Les points d'entrée du logiciel sont mis en place via un pattern Facade. Ainsi, ce sont ces classes qui sont contactées par les services et qui font office de routage du programme. Les façades ont également pour but de vérifier et valider la présence des arguments. Par exemple, l'utilisation du drone est effectuée par l'implémentation d'un pattern facade, ici utilisé par la classe DroneFacade. C'est grâce à cet objet que les services peuvent accéder et modifier les informations des drones. Une fois les données primitives reçues, la façade contacte le manager adéquat avec les bons arguments (droneId:int -> drone: Drone) qui s'occupe de mettre à jour ou retourner les informations des drones. Cependant, si l'identifiant du drone est erroné, la façade se charge d'envoyer une exception.

Pour chaque groupe d'éléments nous avons également mis en place un pattern Singleton qui contient la liste de ces derniers. Ce pattern s'est avéré utile car le fait de stocker les éléments dans un seul singleton permet de centraliser les objets qui sont ainsi facilement accessibles par les classes les demandant. Par exemple, nous disposons d'un singleton contenant la liste des drones. Grâce à ce pattern, le DroneManager ainsi que le DeliveryManager peuvent facilement accéder à la liste des drones. Nous pouvons supposer que le singleton fera office de base de données le temps qu'elle soit mise en place.

Drone

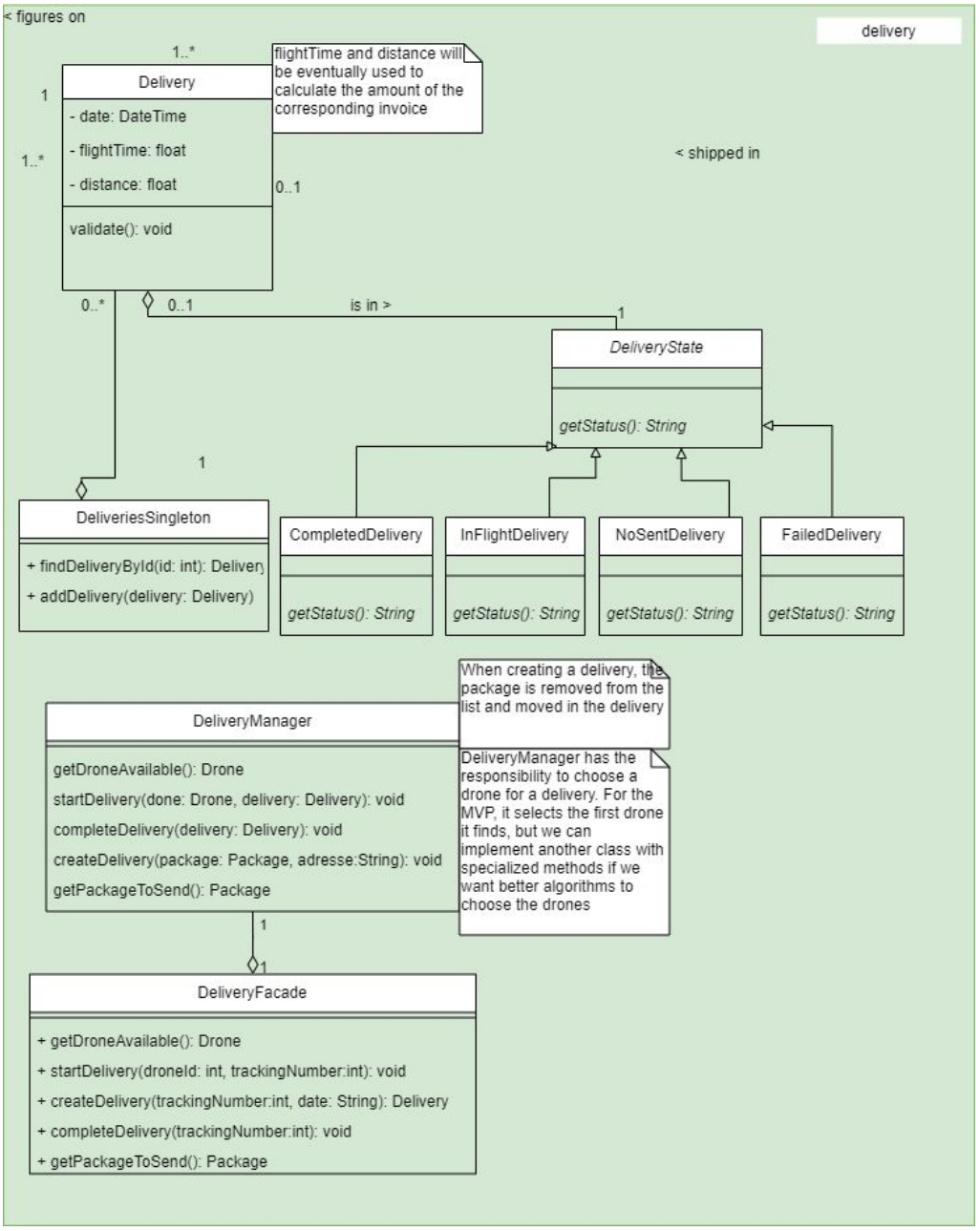
Nous nous sommes dans un premier temps focalisés sur la gestion des drones et les interactions que Livrair pourrait avoir avec ceux-ci. Nous en sommes venus à imaginer le diagramme suivant :



Notre classe Drone représente l'entité d'un drone avec ses divers paramètres. Nous avons mis en place un pattern State pour connaître l'état d'un drone et ainsi grâce à la méthode getRemainingTime() savoir le temps qu'il faut avant que le drone soit à nouveau disponible. L'utilisation d'un tel pattern est intéressant car il permet de déléguer la responsabilité de connaître le temps d'utilisation restant à une seule classe et ainsi empêcher l'utilisation excessive de if.

Nous avons également pensé à utiliser un pattern Factory pour la création du drone. Pour l'instant, Livrair ne dispose que d'un seul type de drone mais nous pouvons imaginer que par la suite, de nouveaux types de drones avec de nouvelles fonctionnalités seront ajoutés. Dans cette optique-là, l'utilisation d'une factory peut être intéressante pour déléguer la création de divers drones à une seule classe.

Delivery

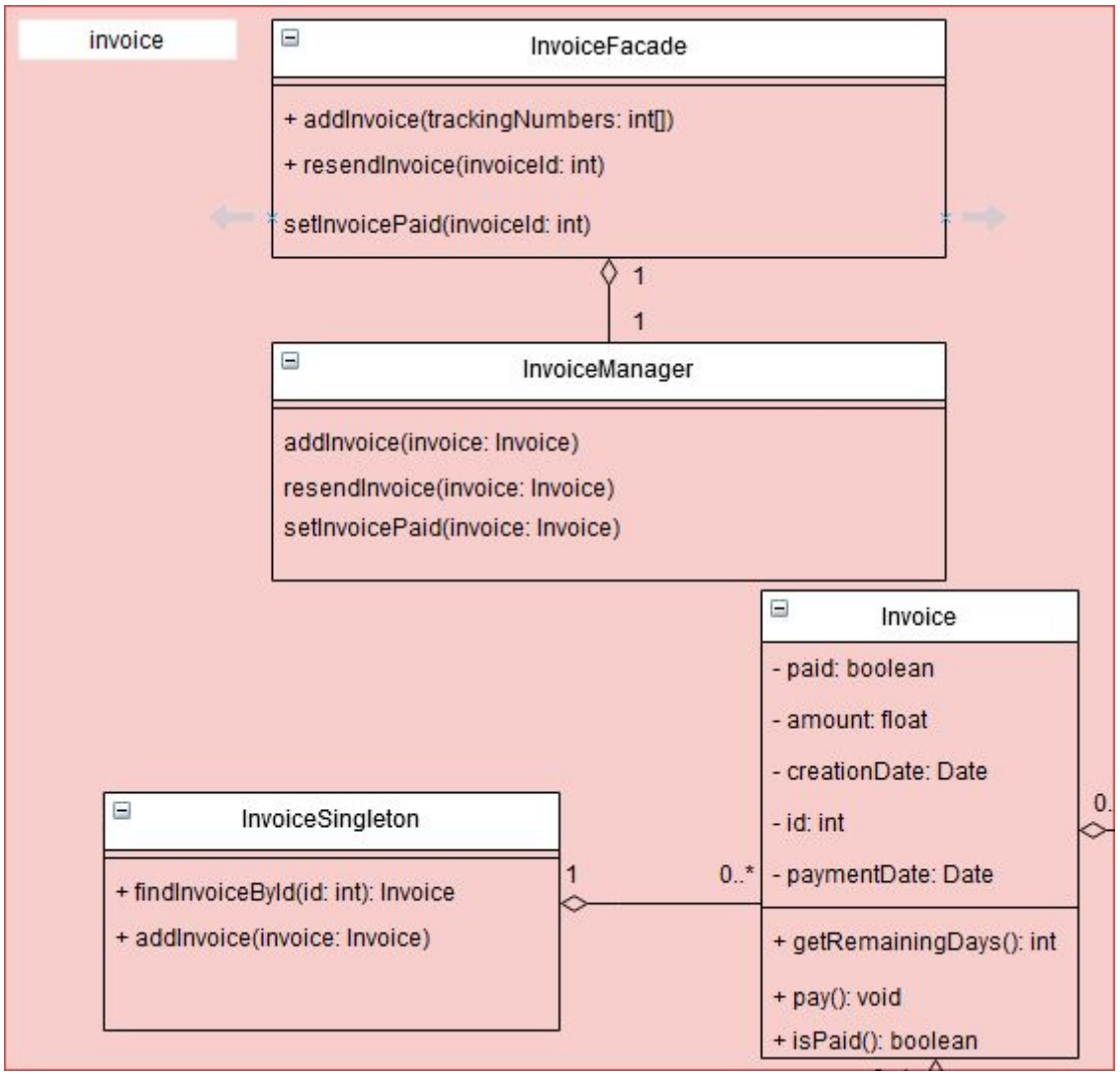


Le service client, après consultation avec le client, créée alors une Delivery avec la méthode createDelivery() en indiquant le numéro du colis ainsi que la date et l'heure d'envoi. Le manutentionnaire a accès aux informations de livraison grâce à la méthode getPackageToSend(). Il peut ensuite commencer les livraisons, et après la livraison du drone, il peut indiquer si la livraison s'est bien déroulée ou si elle a échoué.

Le service client peut accéder aux Delivery ayant échoué, et recontacter le client pour discuter d'une nouvelle date de livraison. En utilisant la méthode createDelivery(), le système se charge, au lieu de créer un nouvel objet, de simplement remplacer la Delivery déjà existante.

Invoice

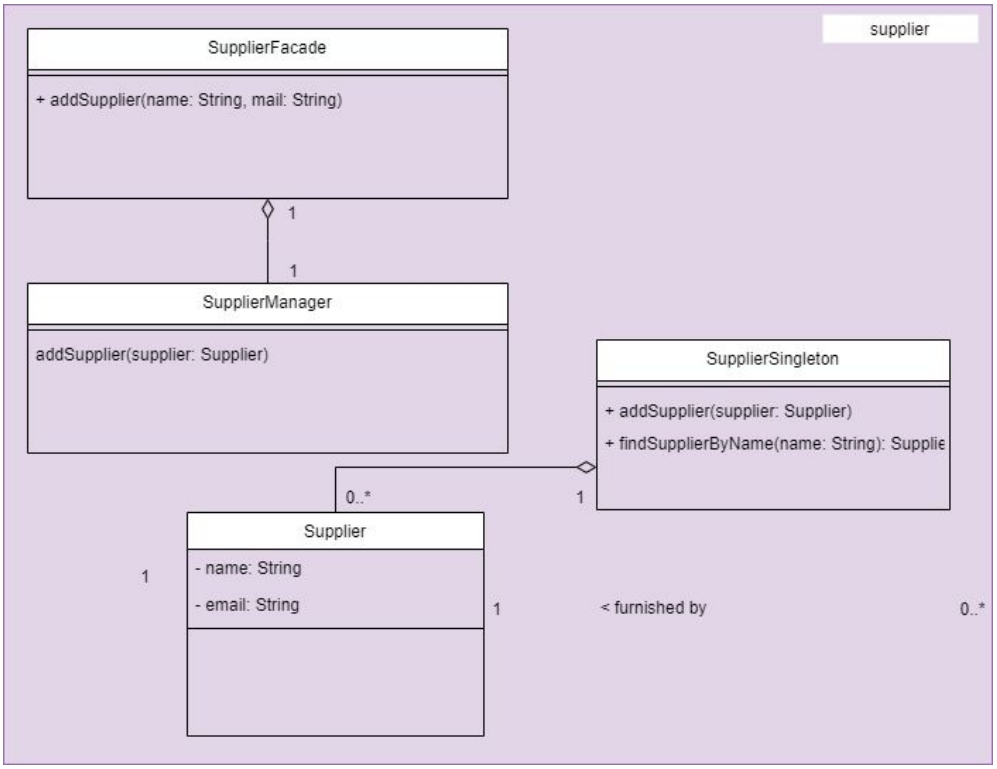
Cette partie du diagramme de classes se focalise sur la gestion des factures par la gestionnaire de l'entreprise.



La InvoiceFacade est la classe d'interaction entre le client et le système. Les objets Invoice sont, dans notre scénario, créés tous les jours automatiquement après une livraison réussie. Chaque Invoice présente le montant facturé, le numéro de la facture (id unique), la liste des Delivery ayant été effectuées, ainsi que la date de création. Toutes les factures (Invoice) doivent être payées par le Supplier sous 30 jours. Si la méthode getRemainingDays() renvoie un nombre inférieur à 0, alors le délai de paiement a été dépassé.

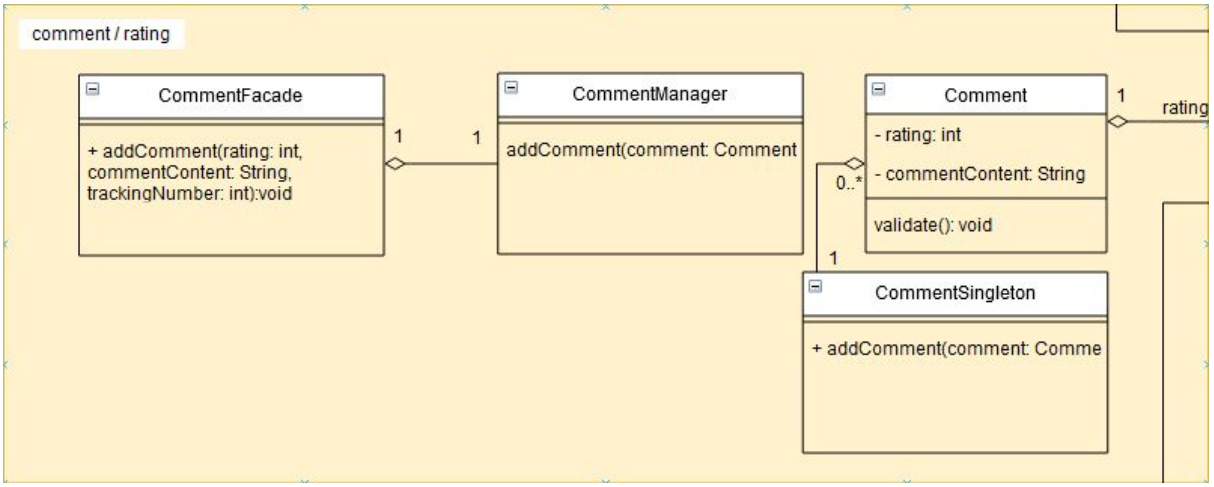
La gestionnaire peut créer des factures depuis la classe InvoiceFacade en entrant les numéros des packages livrés au cours de la journée. L'objet Invoice se remplit automatiquement avec la Delivery attribuée au numéro de package indiqué. Le statut de la facture est indiqué par la méthode isPaid().

Supplier



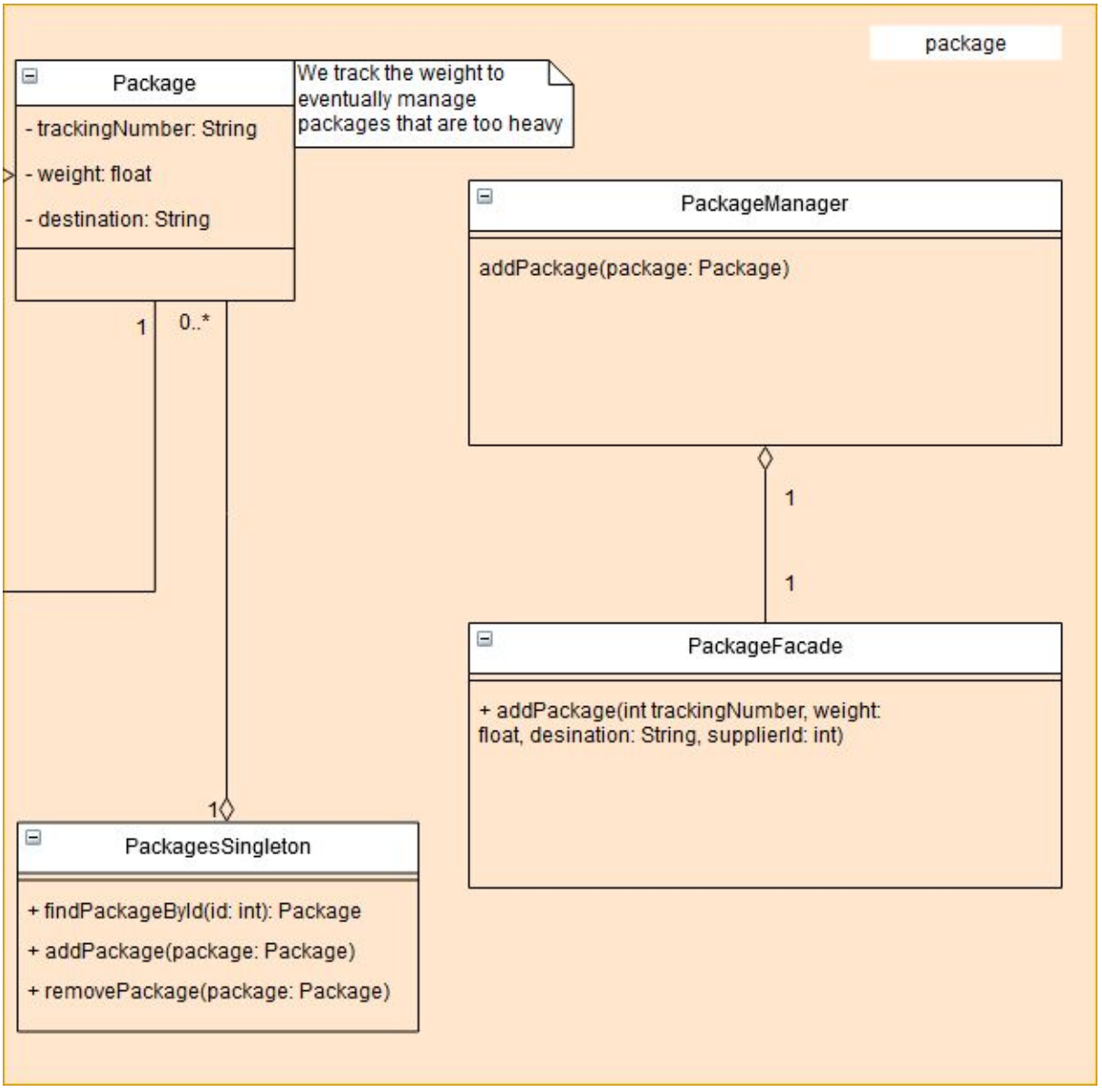
Les objets Supplier représentent les fournisseurs qui livrent les colis à Livrair. Ces objets permettent en fin de mois de lier les factures à des entités, et plus généralement de garder une liste des fournisseurs avec qui Livrair travaille, ainsi que de voir si les livraisons de colis émis par certains fournisseurs ont des commentaires plutôt positifs ou négatifs.

Comment / Rating



Nous avons également imaginé un système de commentaires pouvant être déposés sur une livraison. Les clients peuvent, grâce à leur code de suivis de colis, utiliser une interface, potentiellement web pour faire au plus simple, et ainsi déposer un avis sur une livraison.

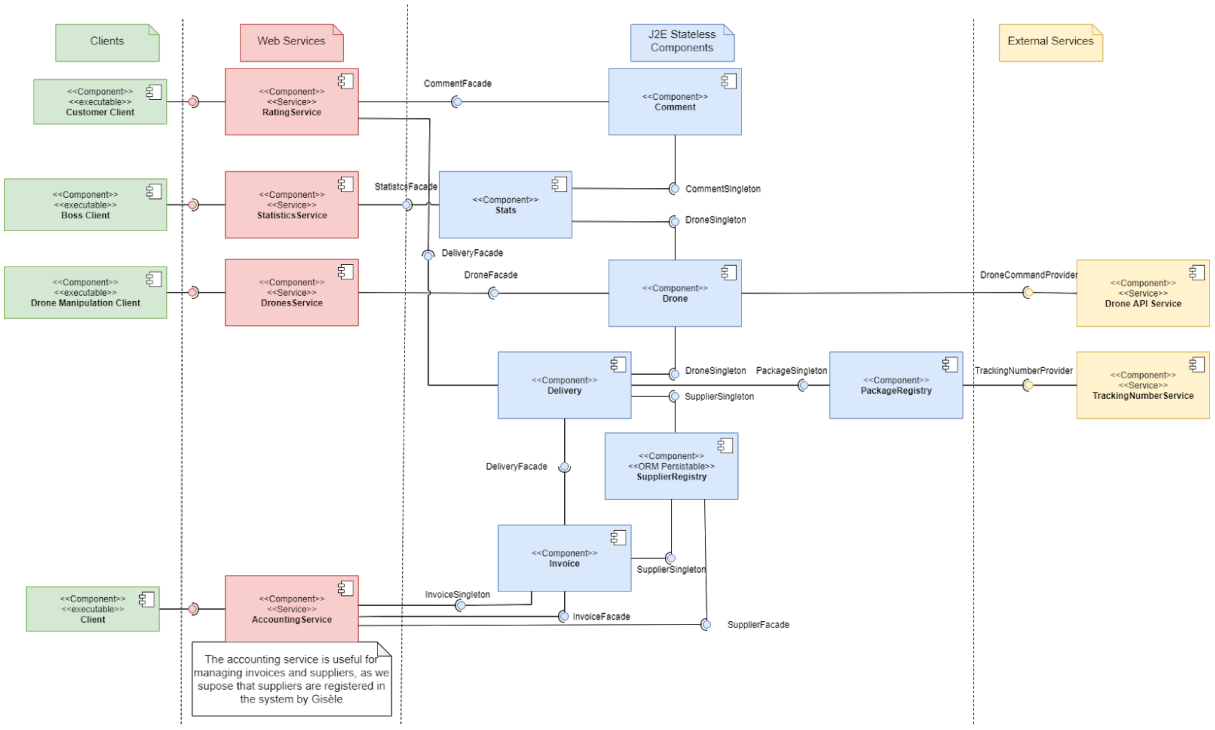
Package



Cette partie représente la gestion des colis dans le système.

La classe Package représente une instance de ce dernier avec son identifiant unique : le numéro de suivi, le poids, la destination ainsi que son expéditeur qui n'est ni plus ni moins qu'un Supplier. Lors de la réception d'un colis, on utilisera PackageManager par l'intermédiaire de PackageFacade pour ajouter le colis dans le système afin qu'il puisse être géré par ce dernier.

Diagramme de composants



Les composants sur lesquels nous allons nous concentrer sont les beans J2E, nous n'allons pas détailler les Web Services dans le cadre du MVP.

Prototypes des interfaces

CommentFacade: Utilisée pour manipuler les commentaires (création) depuis l'interface du client.

```
void createComment(String trackingNumber, String text, int rating);
```

StatisticsFacade: Utilisée pour accéder aux statistiques de l'utilisation des drones et de la satisfaction client depuis l'interface dédiée au boss.

```
float getDronesUseRate();
float getDroneUseRate(int droneID);

float getOverallCustomerSatisfaction();
float getCustomerSatisfactionOnDay(Date day);
float getCustomerSatisfactionSince(Date day);
float getCustomerSatisfactionOnDelivery(String trackingNumber);
```

DroneFacade: Utilisée pour manipuler les drones (ajout, changement d'état) depuis l'interface du manutentionnaire.

```
void changeDroneState(Drone drone, DroneStatusState state);
int addDrone(); // returns automatically-generated drone ID
```

DeliveryFacade: Utilisée pour manipuler les livraisons (ajout, changement d'état, modification,) depuis l'interface du manutentionnaire.

```
void changeDeliveryState(String trackingNumber, DeliveryStatusState
state);
void setDeliveryDate(String trackingNumber, Date newDate);
void setDeliveryDistance(String trackingNumber, double distance);
```

InvoiceFacade: Utilisée pour manipuler les factures (ajout, modification) depuis l'interface de la gestionnaire.

```
int createInvoice(String supplier, double amount); // returns
automatically-generated invoice ID
void payInvoice(int id);
```

SupplierFacade: Utilisée pour manipuler les fournisseurs (ajout, modification) depuis l'interface de la gestionnaire.

```
void createSupplier(String name);
void changeSupplierName(String oldName, String newName);
```

CommentSingleton: Sert de "finder" pour rechercher des commentaires en particulier dans la liste de ceux créés (dans la base de données). On peut rechercher en particulier un commentaire sur une livraison, ou des commentaires en fonction de leur note.

```
Comment findCommentOnDelivery(String trackingNumber);
List<Comment> findCommentsRated(int rating);
```

DroneSingleton (-> Composant Stats): Permet de rechercher les drones en fonction de leur identifiant pour récupérer leurs attributs afin de générer les statistiques.

```
Drone findDrone(int droneId);
List<Drone> getDronesByState(DroneStatusState state);
```

DroneSingleton (-> Composant Delivery): Permet de rechercher les drones en fonction de leur identifiant pour récupérer leurs attributs afin de l'associer à la delivery.

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

PackageSingleton: Permet de trouver les colis dans la base de données.

```
List<Package> getUnsentPackages();
Package getPackage(String trackingNumber);
```

SupplierSingleton: Permet de trouver les fournisseurs dans la base de données.

```
Supplier getSupplier(String name);
List<String> getAllSuppliers();
```

InvoiceSingleton: Permet de trouver les factures dans la base de données.

```
Invoice findInvoice(int id);
List<Invoice> getWaitingInvoices();
List<Invoice> getLateInvoices();
```

Pour visualiser le diagramme de classe dans son ensemble voici un lien draw.io :
https://drive.google.com/file/d/1xIhiTED15StvB9Fz7Q-jAAIijBWXNjI8/view?usp=sharing

Pour visualiser le diagramme de composant voici également un lien draw.io :
https://drive.google.com/file/d/1QdNMC2uXqrskWlzwgJsRvZ-jEKe1Va1N/view?usp=sharing