Drone Delivery

RAPPORT D'ARCHITECTURE

Introduction to **S**oftware **A**rchitecture | **(Philippe Collet)**

>_Team D

DJEKINOU Paul-Marie

KOFFI Paul

MESSAN Aurore

MONIN Donélia

NABAGOU Djotiham

Lundi, 17 février 2020

Sommaire

Introduction

Le métier d'un architecte consiste en l'aptitude de ce dernier à évaluer au mieux toutes les conditions requises pour l'établissement d'un écosystème favorable à l'environnement dédié. Il peut être amené dans sa tâche à effectuer des compromis de conception afin de se rapprocher du meilleur rendu possible. Il est ainsi question dans ce cours d'Introduction à l'Architecture Logicielle de prendre connaissance de ces aptitudes pour pouvoir faire face aux logiciels d'aujourd'hui qui sont plus exigeants autant en fonctionnalités qu'en ressources. Pour appliquer ces aptitudes, il est question cette fois-ci de modéliser un système de livraison de colis par drone afin de pallier aux contraintes de temps que subissent les transporteurs traditionnels. Intitulé « *Drone Delivery* », ce système devra permettre d'automatiser au mieux le processus de livraison des colis aux clients. Ce présent rapport fait mention des principaux diagrammes de modélisation couplés aux justifications qui ensemble, forment l'architecture de notre problème.

I. Diagramme de cas d'utilisation

1. Représentation

Pour des raisons d'espace de représentation, nous répartirons notre diagramme de cas d'utilisation du système « *Drone Delivery* » en 3 parties.

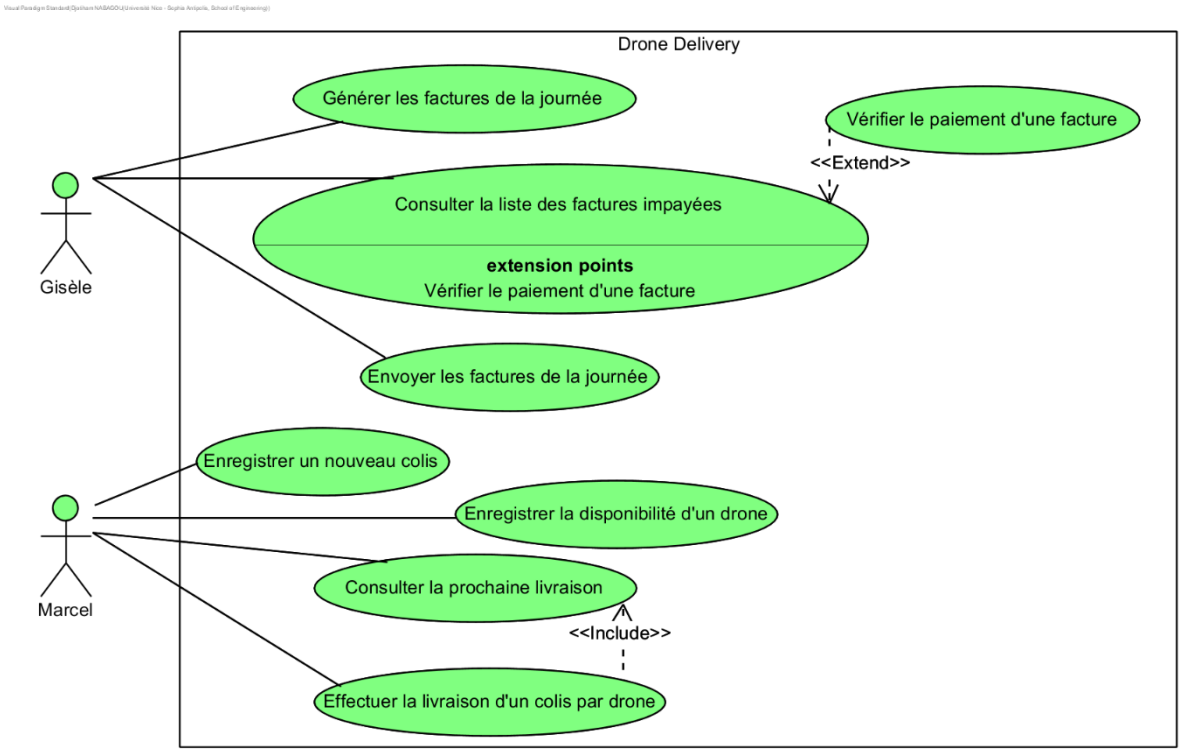

Figure 1 : Diagramme de cas d'utilisation des acteurs Marcel et Gisèle

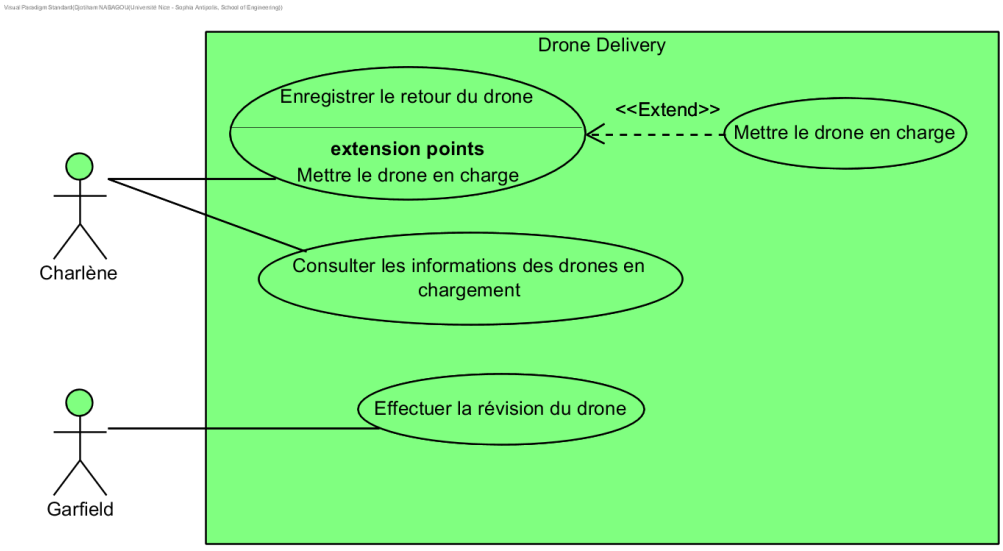

Figure 2 : Diagramme de cas d'utilisation des acteurs Charlène et Garfield

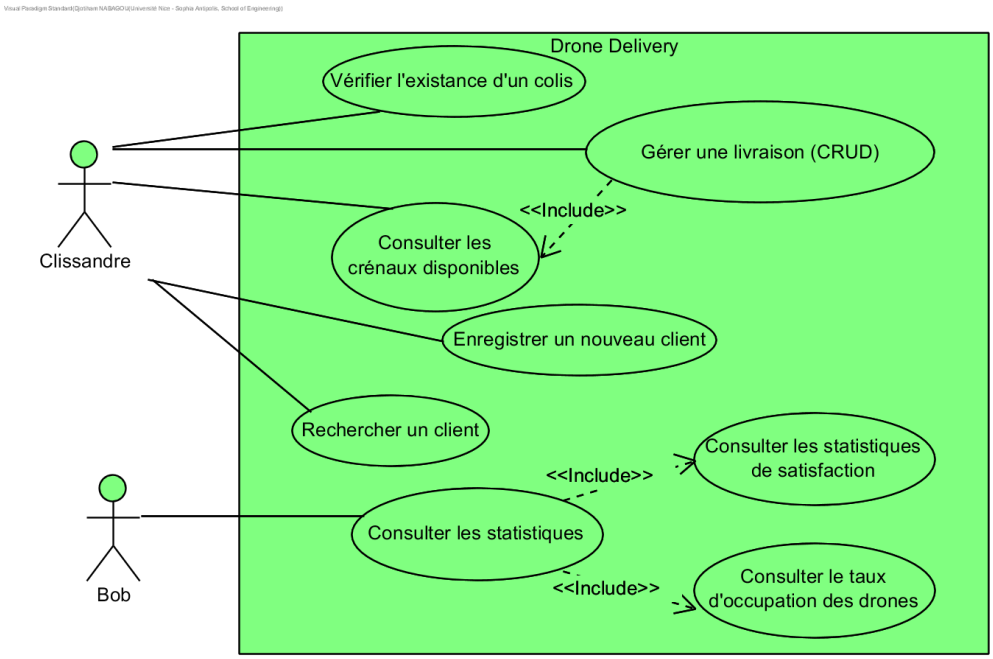

Figure 3 : Diagramme de cas de l'acteur Clissandre et Bob

2. Description des scénarios

Nous commencerons par une exécution du scénario typique pas à pas en dégageant à chaque étape les différentes actions qui seront exécutées par les acteurs de notre système.

- **Un camion de transporteur vient d'arriver**

Marcel doit enregistrer chaque colis livré avec ses références (Id du colis, Id du transporteur, etc…). Pour cela, il réalise donc l'action « **Enregistrer un nouveau colis** ».

- **Le client appelle Clissandre pour prévoir un horaire de livraison**

Cette action (appel) est effectuée en dehors de notre système. Le but de l'appel étant de prévoir un horaire de livraison qui arrange les 2 parties, Clissandre effectuera donc l'action « **Consulter les créneaux disponibles** ». Elle pourra ainsi par la suite proposer un créneau disponible pour la livraison au client qui peut l'accepter ou non. Il suffira donc qu'elle se mette d'accord avec le client sur l'un des créneaux disponibles. Avant tout ça, Clissandre doit d'abord vérifier l'existence du colis du client, elle effectuera donc l'action « **Vérifier l'existence d'un colis** ». Il ne sert à rien de prévoir la livraison d'un colis qui n'existe pas ou dont Livrair n'est pas encore en possession. Aussi Clissandre doit pouvoir « **Enregistrer un nouveau client** » ou « **Rechercher un client** » (elle vérifiera que les informations concordent bien avec ceux données par le client actuel si celui-ci nous dit qu'il avait déjà reçu une livraison de notre part).

- Consulter les créneaux disponibles : les créneaux disponibles dans notre système seront proposés en tenant compte du statut de tous les drones du système.
- Vérifier l'existence d'un colis : le colis sera vérifié par rapport aux informations données par le client à Clissandre. (Notamment l'id ou le numéro du colis qui a été fournis par le transporteur au client)

- **Clissandre prévoit le créneau dans le planning**

Clissandre effectuera donc ici l'action « **Ajouter une livraison** » pour mettre à jour notre agenda de livraison. Le statut du drone choisi par le système pour la livraison passera donc de Disponible à En attente de livraison.

Les 2 précédents points du scénario typique se résumeront à la description détaillée du cas d'utilisation Ajouter une livraison (nous ne ferons pas la description détaillée totale)

- Nom : Ajouter une livraison
- Description : Un client appelle Clissandre pour prévoir un horaire de livraison
- Conditions préalables : Le colis du client a été livré par le transporteur, le système propose à Clissandre des créneaux disponibles
- Post Conditions : La livraison est enregistrée
- Acteurs : Clissandre

- **Marcel prend un drone disponible et chargé, lui attache le colis et l'envoie**

Marcel peut consulter les détails de la prochaine livraison via l'action **consulter la prochaine livraison** qui affichera l'identifiant de la livraison, l'heure, le numéro du drone chargé de la livraison ainsi que l'identifiant du colis. Il ne lui restera plus qu'à effectuer l'action **effectuer la livraison d'un colis par un drone** pour effectivement démarrer la livraison du drone.

Cela implique que notre système ne proposera jamais à Clissandre un même horaire pour la livraison de 2 ou plusieurs colis même si nous possédons à cette heure 2 ou plusieurs drones disponibles. Marcel ne peut pas faire ces actions simultanément et doit avoir une marge de temps entre 2 livraisons successives (exemple : 5 minutes).

Effectuer la livraison d'un colis par un drone : Le système aura juste besoin de l'identifiant de la livraison prévue (que marcel va entrer) et se chargera de donner au drone qui est associé à la livraison, le signal de départ. Pour une meilleure automatisation des tâches, nous devons réfléchir sur 3 options possibles :

- Marcel démarre à l'heure prévue, marcel voit sur son écran « livraison en cours », le statut du drone qui effectue la livraison passe à En livraison
- Marcel démarre la livraison avant l'heure prévue, marcel voit sur son écran « attendez l'heure de livraison prévue »
- Marcel démarre la livraison après l'heure prévue, plusieurs options s'offrent à nous :
  Nous décidons que chaque client qui programme une livraison chez nous à une heure X est au courant que la livraison interviendra dans la tranche horaire [X, X+5 minutes], ce qui laisse une marge de 5 minutes de retard au plus à marcel. Si la prochaine livraison (heure de départ du drone) est prévue pour 7h et que marcel fait l'action à 7h4, la livraison peut toujours être effectuée (statut du drone changé à En livraison) mais s'il lance l'action à 7h6 par exemple, il doit recevoir le message « livraison annulée, veuillez détacher le colis et replacer le drone ». Le statut du drone sera donc modifié par le système (passe de En attente de livraison à Disponible pour livraison). Á long terme, on peut imaginer que marcel va recevoir une pénalité pour chaque retard occasionné dans sa tâche.

- **Au retour du drone, Charlène le récupère, et soit le met en charge, soit l'apporte à Garfield pour révision, soit l'apporte à Marcel pour la mission suivante. Le retour du drone vide (donc colis livré) déclenche une écriture comptable qui indique à Gisèle que la livraison a bien eu lieu.**

Charlène le récupère sous-entend que cette action est physique et ne se fait pas dans notre système.

Á la récupération d'un drone, Charlène doit faire l'action : « **Enregistrer le retour d'un drone** » (où elle va entrer le numéro du drone, l'heure de retour de ce dernier et le numéro du colis s'il n'a pas été livré). Á la suite de cette action, Charlène verra un message sur l'écran qui sera soit :

- **Amener le drone à Marcel**
- **Mettre le drone en charge**
- **Mettre le drone en réparation**

Le système va envoyer le message en fonction de notre algorithme de répartition des tâches aux drones (planning de livraison) et donc des prochaines livraisons (ou non) du drone tout en tenant compte des différentes contraintes de vol d'un drone (révision + charge). Si le colis a été livré avec succès, le système va modifier le statut de la livraison. On suppose aussi que dans le cas où la livraison n'est pas effectuée, Charlène rapporte le colis à Marcel qui le range dans l'une des étagères à nouveau.

Si Charlène doit mettre le drone en charge, le statut du drone pendant ce temps est En attente de chargement. Charlène met le drone en charge et fait l'action « **mettre le drone en charge** » (qui prendra en paramètre le numéro du drone et l'heure début chargement) pour que le système mette à jour le statut du drone à En chargement. Elle ramènera le drone à marcel lorsque celui-ci sera chargé. Elle peut donc faire l'action « **consulter les informations des drones en chargement** » pour savoir le temps restant ainsi que l'heure de fin de chargement pour chaque drone. Lorsque l'heure de fin de chargement sera atteinte, le drone passera automatiquement au statut En attente de récupération.

Si le drone doit être révisé, son statut passe automatiquement à En attente de réparation et Charlène amène le drone à Garfield (action physique). Garfield avant de commencer la révision du drone doit faire l'action « **Effectuer la révision du drone** » (pour notifier au système qu'il vient de commencer effectivement la révision du drone dont il va passer le numéro en paramètre). Le statut du drone va donc passer à En réparation. Garfield ne peut pas réviser 2 drones simultanément. Il faudra en tenir compte, c'est pourquoi il faut que Garfield nous indique lui-même le début de la révision car il se peut qu'il y ait une liste d'attente de révision ce qui implique que la révision d'un drone ne démarrera pas forcément lorsque celui-ci sera amené par Charlène. Lorsque l'heure de fin de réparation sera atteinte, le drone passera automatiquement au statut En attente de récupération.

Si Charlène a plutôt reçu le message « Amener le drone à Marcel », le statut du drone sera En attente de récupération.

Enfin, lorsque Marcel reçoit un drone de la part de Charlène (drone venant de terminer une livraison ou drone chargé) ou de Garfield (drone réparé), son statut est forcément En attente de récupération. Il effectue donc l'action « **Enregistrer la disponibilité d'un drone** » qui fera passer le statut du drone à Disponible pour livraison.

Aperçu des différents états d'un drone

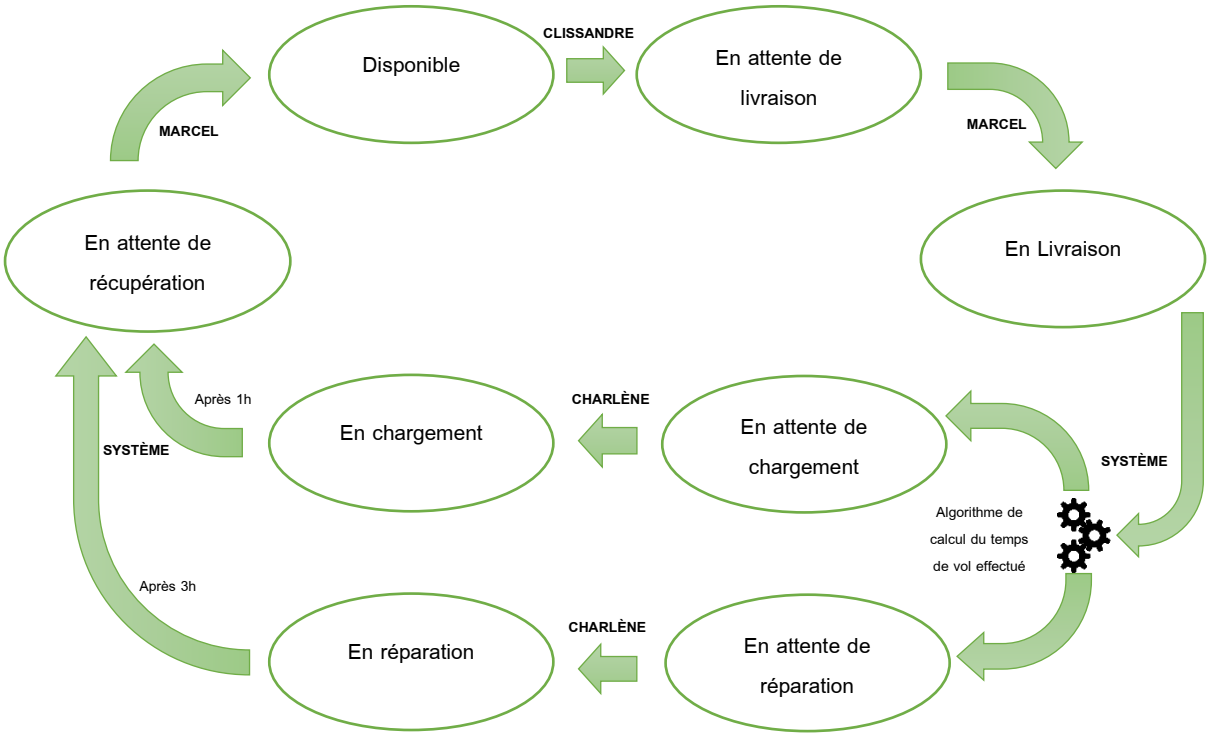

Figure 4 : Cycle d'état d'un drone

- **Tous les soirs, Gisèle émet des factures**

L'envoi des factures aux différents transporteurs ne se fera pas dans notre système à court terme. Néanmoins à long terme, nous pensons à automatiser ce processus d'envoi. Stockant dans notre base de données les informations concernant toute les livraisons (notamment dateLivraison, transporteur, prixLivraison). Nous pouvons automatiser la génération de toutes les factures de la journée. On pense notamment à l'utilisation d'un package JAVA existant permettant la création de PDF. Le système va donc générer pour chaque transporteur (dont nous avons livré au moins un colis dans la journée) un fichier PDF qui sera sa facture du jour. Gisèle aura donc l'action « **Générer les factures de la journée** » qui va lui retourner la liste des factures générées. Un exemple du résultat de cette action sera :

TRANSPORTEUR	MAIL	FACTURE
FEDEX	fedex@gmail.com	fedex_05/05/2020.pdf
UPS	ups@gmail.com	ups_05/05/2020.pdf

L'envoi des factures étant hors de notre système à court terme, Gisèle va donc envoyer la facture PDF générée pour un transporteur à son adresse email. Elle aura en plus l'action « **Consulter la liste des factures impayées** » (qui lui retournera par exemple la liste des factures groupées par transporteur ainsi que leurs références). Parmi cette liste elle peut effectuer l'action « **Vérifier paiement d'une facture** » (en fournissant la référence de la facture) qui va permettre de mettre à jour le statut d'une facture donnée et qui va retourner un résultat SOLDÉ / IMPAYÉE. Á long terme nous pensons à une action « **Envoyer les**

factures de la journée » qui fournira le même rendu que ci-dessus (action générer les factures) mais qui en plus, va envoyer automatiquement les factures aux transporteurs à leurs mails respectifs.

- **En continu, Bob suit les indicateurs métiers**

Bob quant à lui peut effectuer 2 actions : « **Consulter les statistiques de satisfaction** » qui lui afficheront la moyenne de toutes les notes de satisfaction (et peut être plus de détails) ainsi que « **Consulter le taux d'occupation des drones** » qui affichera plutôt les statistiques sur les drones (drones en chargement, en réparation, en livraison, etc…)

II. Diagramme de composants

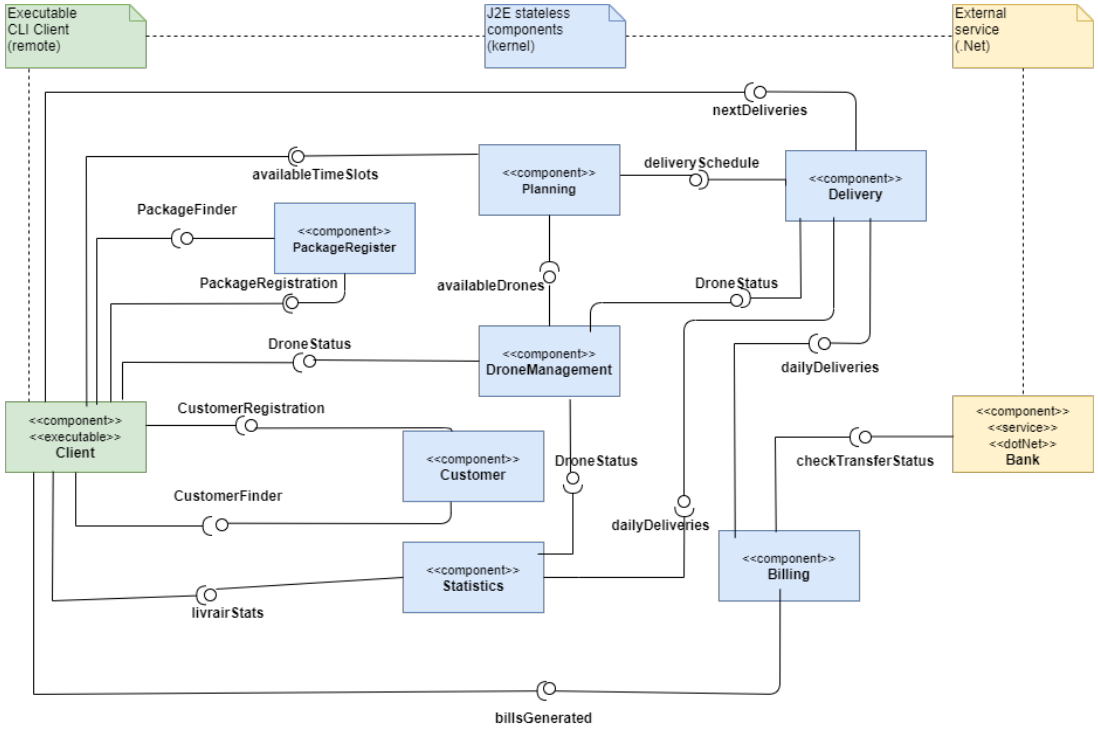

Figure 5 : Diagramme de composants

III. Définition du pseudocode des interfaces

1. Interface PackageFinder

findById(String) : Optional<Colis> prend en paramètre un id de colis et retourne un colis s'il est trouvé

2. Interface PackageRegistration

register(String, double, date) : void prend en paramètre un id de colis, le poids du colis et la date du jour , méthode permettant d'enregistrer un nouveau colis

3. Interface CustomerFinder

findByName(String) : Optional<Client> prend en paramètre le nom de client et retourne un client s'il est trouvé

4. Interface CustomerRegistration

register(String, String, String) : void prend en paramètre le nom, le prénom et l'adresse du client (va générer automatiquement un id pour le client)

5. Interface AvailableTimeSlot

getAllSlot() : void va retourner une liste de plages horaires disponibles

verifySlot(String, String) : boolean prend en paramètre la date et l'heure (sous forme de string, les formatages éventuels seront faits à l'intérieur de la méthode) et permet de vérifier directement une plage horaire voulue par le client

6. Interface AvailableDrones

getAvailableDrones(String, String) : Optional<Drone> prend en paramètre la date et l'heure (sous forme de string, les formatages éventuels seront faits à l'intérieur de la méthode) et va retourner la liste des drones disponibles sur cette plage horaire

7. Interface DeliverySchedules

getDeliverySchedules() : Optional<Livraison> va retourner la liste des livraisons qui sont déjà prévues

8. Interface CheckTransferStatus

getTransferStatus() : void va retourner la liste des factures payées

getTransferStatus(String) : void va prendre en paramètre l'id d'une facture et va retourner la liste des factures payées

9. Interface NextDeliveries

getNextDeliverie() : Optional<Livraison> va retourner les informations sur la prochaine livraison

10. Interface DailyDeliverie

getAllDayDeliverie(String) : Optional<Livraison> va renvoyer la liste de livraison du jour en fonction de l'id du transporteur passé en paramètre

11. Interface DroneStatus

getAllLoadingDrone() : Optional<Drone> va renvoyer la liste des drones en chargement ainsi que les informations nécessaires

getAllFixingDrone() : Optional<Drone> va renvoyer la liste des drones en réparation ainsi que les informations nécessaires

setStatusDrone(String,status,Heure) : void va modifier le statut du drone (sera utilisé pour indiquer le début de chargement et de réparation) d'un drone

12. Interface LivrairStats

printSatisfactionCustomerStats(String, String) va renvoyer la moyenne des satisfactions sur la période donnée

printDroneStats(String, String) va renvoyer les statistiques sur les temps d'occupation des drones

IV. Diagramme de classe

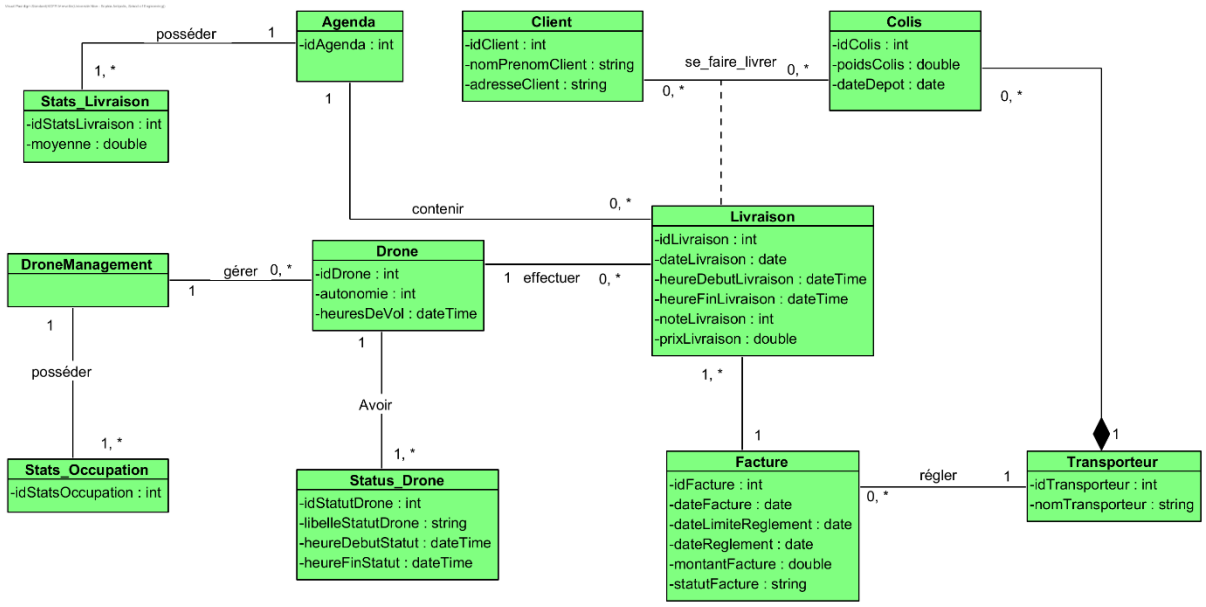

Figure 6 : Diagramme de classe Drone Delivery

Attardons-nous sur la relation entre Client et Colis : on peut supposer dans un premier temps qu'il apparait entre un client et un colis une relation d'ordre **1---0…*** (où Client posséderait donc la liste de colis qui lui sont destinés). Cette approche n'est pas vraiment viable car cela impliquerait qu'il y'ait une relation d'ordre **1---0…*** entre Client et Livraison (**ce qui n'est pas raisonnable car pour une livraison, on doit avoir une trace du colis qui a été livré**). **Si plusieurs colis pouvaient être livrés en une livraison**, la meilleure conception serait d'établir une relation d'ordre **1---1…*** entre Livraison et Colis, et **1---0…*** entre Client et Livraison. Dans notre contexte, la meilleure solution est la création de la table Livraison en tant que classe association entre un client et un colis destiné à ce client.

Pour des raisons de sécurité également, quand un client appelle pour programmer une livraison, il devrait d'abord nous fournir (donc à clissandre) l'id du colis (numéro secret qu'il serait seul à connaitre) dont il est question.

Un transporteur livre plusieurs colis à une date donnée. Une facture à une date donnée pour un transporteur connu est constituée de l'ensemble des livraisons (aux clients) de colis provenant de ce transporteur à cette date. Un transporteur peut recevoir 0 ou plusieurs factures en fonction des livraisons de colis de la journée. Un drone peut effectuer plusieurs livraisons et possède à une heure donnée un seul statut. Pour retracer l'historique des mouvements d'un drone, il suffit donc de consulter la liste de tous les statuts du drone en question pour la journée choisie. On pourra donc savoir avec précision les opérations de notre flotte de drone (les moments de la journée où un drone était en charge, en réparation, en livraison, etc…). Cela est indispensable pour des statistiques d'occupations précises.

Pour le planning des livraisons, il faut une classe nous permettant l'ajout, la modification ou la suppression d'une livraison. De plus cette classe doit nous permettre de reconstituer l'historique de toutes les livraisons (effectuées avec succès ou pas). Elle intervient aussi en partie dans la recherche de nouveaux créneaux disponibles. Enfin elle permet de générer des statistiques de satisfaction qui découleront des notes attribuées par un client pour une livraison. Cette classe est représentée par Agenda sur le diagramme de classe.

Enfin la classe DroneManagement quant à elle est celle responsable de la gestion de la flotte de drone. Elle se chargera du changement automatique de statut d'un drone ainsi que de toutes les opérations (avoir la liste des drones disponibles en fonction d'un horaire par exemple, consulter la liste des drones en chargements, vérifier la liste des drones en réparations, etc…). C'est cette classe également qui nous permettra d'établir à la fin des statistiques poussées sur l'activité de la flotte de drone.

Conclusion

Ce rapport consiste en la première phase de notre projet qui est sans doute la plus importante car elle seule nous garantit de construire un système cohérent et surtout qui repose sur une architecture. Grâce au scénario typique qui nous a été fourni dans la spécification, nous avons pu dégager réellement les actions qui se produisaient à chaque étape par chacun des acteurs concernés du système. Actions connues dans leurs déroulements principaux et parfois alternatifs, nous avons pu ensuite identifier celles qui impactent directement sur le système comme étant nos cas d'utilisation. La principale difficulté aura été d'élaborer un diagramme de composants cohérent et répondant le plus possible au scénario typique qui nous a servi de référence. Le diagramme de classe et la définition des interfaces en pseudo-code n'aura été qu'une conséquence du diagramme de composants, le tout formant notre architecture.