Architecture Logicielle

Variante 9 : Mobile App for Transporters

Groupe I

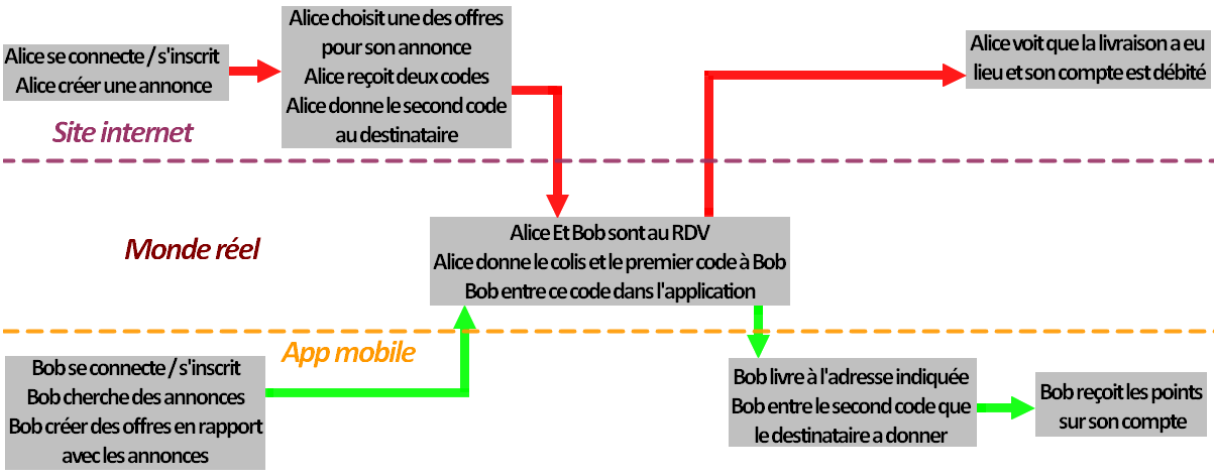
Duminy Gaétan, Picard Marchetto Ivan & Ritrovato Dylan

User Story

PREMIER CYCLE DE DEVELOPPEMENT

Alice (Client)	Bob (Transporteur)
Phase 1a -Alice s'inscrit / se connecte sur le site internet -Alice peut consulter son solde de points -Alice créer son annonce en indiquant le point de départ, le point d'arrivé, les objets à transporter et une fourchette pour la date. Le système calcule le coût en points de l'intervention	Phase 1b -Bob télécharge l'application Android -Bob s'inscrit / se connecte sur l'app -Bob peut consulter son solde de points -Bob lance une recherche d'annonces en indiquant, sa ville de départ, sa ville de destination et la taille maximal du bagage à transporter -Bob ajoute une annonce à sa liste de transport -L'application indique à Bob que son coffre n'est pas rempli en lui indiquant l'espace restant et qu'il peut encore ajouter des annonces à sa liste -Bob peut supprimer une ou plusieurs annonces de sa liste de transport -Bob valide son panier en indiquant ses disponibilités pour chaque annonce
Phase 2 -Alice reçoit des offres à son annonce, proposés par plusieurs transporteurs -Alice choisit l'offre de Bob -Alice reçoit deux codes par mail, un pour la preuve de dépôt et un pour la preuve de réception	
Phase 3a -Alice vient au RDV avec ses objets et rencontre Bob -Alice donne son code de dépôt à Bob	Phase 3b -Bob vient au RDV avec son véhicule et rencontre Alice -Bob entre le code de dépôt que Alice lui a donné
	Phase 4 -Bob livre, à l'adresse indiquée, les objets de Alice (cela peut-être à Alice, une personne désignée par Alice ou bien un autre livreur dans le cas d'un relais) -Bob entre le code réception que le destinataire lui a donné
Phase 5a -Alice peut voir la preuve de reçu sur le site internet -Le compte de Alice se fait débiter du nombre de points associé à sa commande -Alice peut consulter son historique de commande	Phase 5b -Bob reçoit ses points sur son compte -Bob peut consulter son historique de contrats

Remarque : La user story de Alice sera mocké car elle n'est pas nécessaire à l'application en elle-même.



Vue simplifiée des User Story

SECOND CYCLE DE DEVELOPPEMENT

Dans le cas du second cycle de développement, Bob devient un transporteur particulier. Notre nouveau persona se nomme Cédric, Cédric est un transporteur professionnel.

La User Story de Cédric sera ressemblante sur certains points à celle de Bob du fait que certaines interactions reste les mêmes.

Alice **(Client)**	**Bob** **(Transporteur particulier)**	**Cédric** **(Transporteur professionnel)**
Phase 1a -Alice s'inscrit / se connecte sur le site pour les clients -Alice peut consulter son solde de points -Alice créer son annonce en indiquant le point de départ, le point d'arrivé, les objets à transporter et une fourchette pour la date. Le système calcule le coût en points de l'intervention	Phase 1b -Bob télécharge l'application Android -Bob s'inscrit / se connecte sur l'app -Bob peut consulter son solde de points -Bob lance une recherche d'annonces en indiquant, sa ville de départ, sa ville de destination et la taille maximal du bagage à transporter -Bob ajoute une annonce à sa liste de transport -L'application indique à Bob que son coffre n'est pas rempli en lui indiquant l'espace restant et qu'il peut encore ajouter des annonces à sa liste -Bob peut supprimer une ou plusieurs annonces de sa liste de transport -Bob valide son panier en indiquant ses disponibilités pour chaque annonce	Phase 1c -Cédric s'inscrit / se connecte sur l'API pour les professionnels -Cédric est redirigé vers la page d'accueil où il peut visualiser son solde et les différents liens -Cédric clique sur le lien de recherche d'annonce, il a donc accès à un formulaire pour trier les annonces qui correspond à ces critères et accède à une liste d'annonces - Cédric créer un CSV contenant les id des annonces qu'il souhaite ainsi que les dates de disponibilité - Cédric upload son CSV sur l'API
Phase 2 -Alice reçoit des offres à son annonce, proposés par plusieurs transporteurs -Alice choisit l'offre de Bob ou Cédric -Alice reçoit deux codes par mail, un pour la preuve de dépôt et un pour la preuve de réception		
Phase 3a -Alice vient au RDV avec ses objets et rencontre Bob -Alice donne son code de dépôt à Bob	Phase 3b -Bob vient au RDV avec son véhicule et rencontre Alice -Bob entre le code de dépôt que Alice lui a donné directement dans l'app	Phase 3c -Cédric télécharge le CSV de dépôt contenant les numéros des contrats qui correspondent à son parcours -Cédric vient au RDV avec son véhicule et rencontre Alice -Cédric entre le code de dépôt que Alice lui a donné dans le CSV -Une fois tous les objets récupérés, Cédric envoi le CSV sur l'API Web Pro
	Phase 4b -Bob livre, à l'adresse indiquée, les objets de Alice (cela peut-être à Alice, une personne désignée par Alice ou bien un autre livreur dans le cas d'un relais) -Bob entre le code de réception que le destinataire lui a donné sur l'app	Phase 4c -Cédric télécharge le CSV de réception -Cédric livre objets de Alice -Cédric ajoute le code de réception que le destinataire lui a donné dans le CSV sur la même ligne que le contrat -Une fois tous les objets livrés, Cédric envoi le CSV
Phase 5a -Alice peut voir la preuve de reçu sur le site internet -Le compte de Alice se fait débiter du nombre de points associé à sa commande -Alice peut consulter son historique de commande	Phase 5b -Bob reçoit ses points sur son compte -Bob peut consulter son historique de contrats	Phase 5c -Cédric reçoit ses points sur son compte - Cédric peut consulter son historique de contrats

Interprétation des nouveaux besoins de la seconde itération

On va décomposer les besoins de chacun des nouveaux besoins pour en déduire l'impact sur notre architecture.

1/ Transporteurs professionnels avec des camions, prennent plusieurs colis.

- Désormais un compte à deux statuts possibles, professionnel ou bien particulier.
- Notre système prend déjà en compte le fait qu'un véhicule puissent prendre plusieurs colis.

2/ N'utilisent pas l'app mobile, mais une API pour charger/décharger les colis par lot (via un csv).

- Création d'une API, de préférence accessible sur le plus d'appareils possible. Une web API serait donc préférable.
- Ajout d'un parceur de CSV à l'API serveur pour traiter les fichiers CSV
- Un transporteur professionnel peut récupérer les CSV de dépôt et ceux de réception sur l'API professionnel

3/L'appli mobile doit être à jour. Deux clients possible (API + Appli) sur les mêmes données, l'affichage doit être cohérent.

- Ajout d'une vérification afin de vérifier le statut d'une annonce lors de son ajout de la validation du panier ainsi que lors de la création de l'offre dans l'application Android
- Vérification de chaque offre présente dans le CSV, envoi d'un avertissement avec le CSV à jour (sans les annonces expirées) et demande de validation du nouveau CSV de la part du transporteur.

Le parceur de CSV sera directement intégré dans l'API et non dans un module à part dû à la simplicité de décomposition d'un fichier CSV. Dans le cas d'un fichier plus difficile à décrypter, on aurait pu en faire un module à part entière ou bien pour anticiper ce genre de changement.

Un avantage notable de cette solution est que le transporteur professionnel gagne en rapidité du fait qu'il entre juste le code sur une ligne d'un CSV puis qu'il envoi le fichier une fois son chargement/déchargement terminé. Cependant, ce système possède le défaut de perdre en sécurité. Là où le transporteur particulier entre le code en direct et s'assure de la validité de celui-ci, le transporteur professionnel n'a pas le luxe de cette vérification. Si un client donne un mauvais code, de manière attentionnée ou non, un conflit devra être gérer.

Exemple de fichier CSV de chargement :	**Exemple de fichier CSV de déchargement :**
Numéro contrat, numéro dépôt	Numéro contrat, numéro réception
« 112526 », « 1654164048 »	« 112526 », « 2456464532 »
« 116314 », « 2678549606 »,	« 116314 », « 8952462423 »,
« 141658 », « 4564848646 »	« 141658 », « 9242642642 »
« 142262 », « 7454348489 »	« 142262 », « 0019854794 »

À la suite de la réunion du vendredi 18 Janvier, cette interprétation a changé.

Un CSV qui sera implémenté est celui permettant la création des offres.

On va tout de même garder les deux autres CSV afin de limiter un maximum les accès à l'API, cette option est envisageable car on part du principe où les codes sont toujours donner par les clients et que ces codes sont toujours juste.

Exemple de fichier CSV de création d'offres :

Numéro annonce, Date de récupération proposée
« 112526 », « 21/01/19 11:40 »
« 116314 », « 21/01/19 12:30 »,
« 141658 », « 24/01/19 10:00 »
« 142262 », « 24/01/19 14:00 »

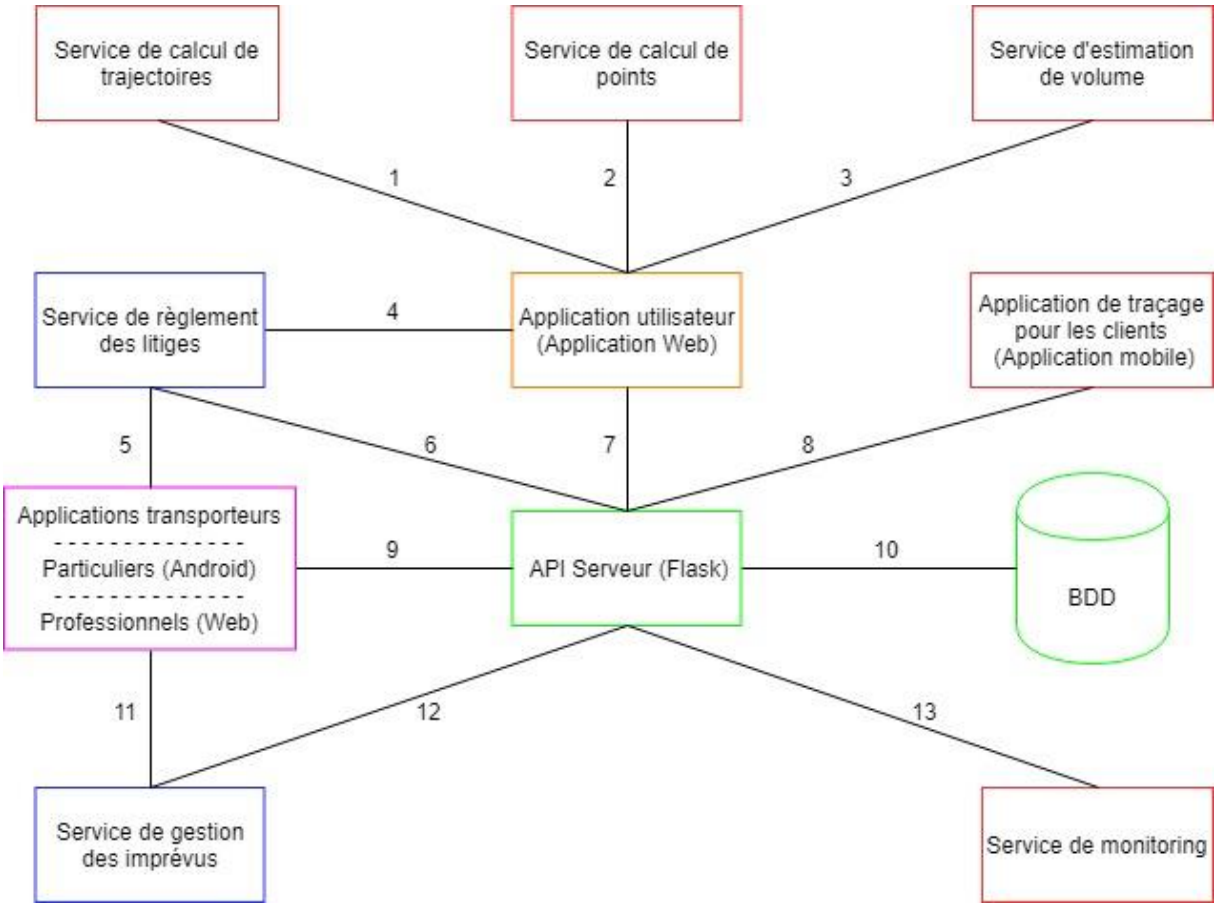
Les autres parties de l'architecture sont adaptées à ce changement.

Mise à jour de la semaine 06 :

Après une réflexion plus importante, il paraît tout de même bien plus viable de remplacer le système de CSV pour les codes de vérification (chargement & déchargement) par un bouton permettant d'entrer le code correspondant au contrat.

La cause n'est pas la mise en place de l'algorithme (facile à produire car un similaire à déjà était codé auparavant), mais tout simplement dû au fait que cela reste plus logique pour un utilisateur.

Diagramme de composants global



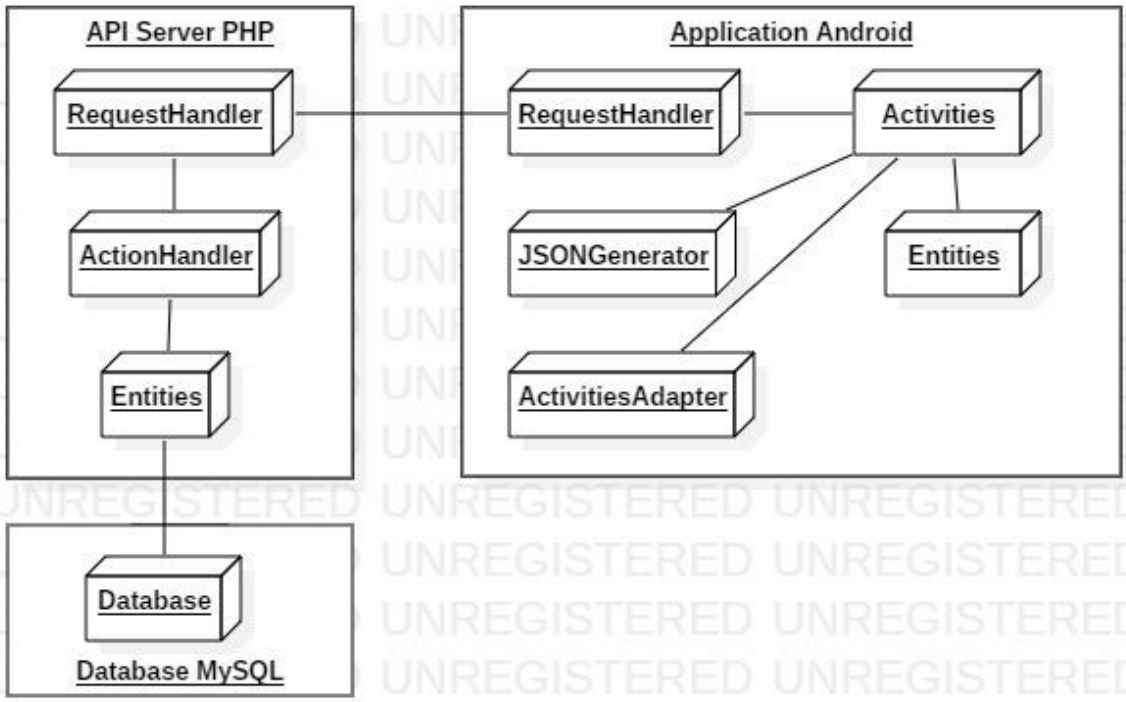
Légende :

Les différentes liaisons représentent une connexion ou des échanges de données entre les deux nœuds correspondants. Les éléments de composants encadrés en vert sont ceux à implémenter, les oranges sont ceux à mocker pour le POC du 9/11/18, les bleus sont ceux à mocker pour le POC du 15/02/18 et les rouges sont ceux qui n'entre pas dans notre scope. Le composant en violet représente en réalité deux composants à implémenter (second cycle de développement).

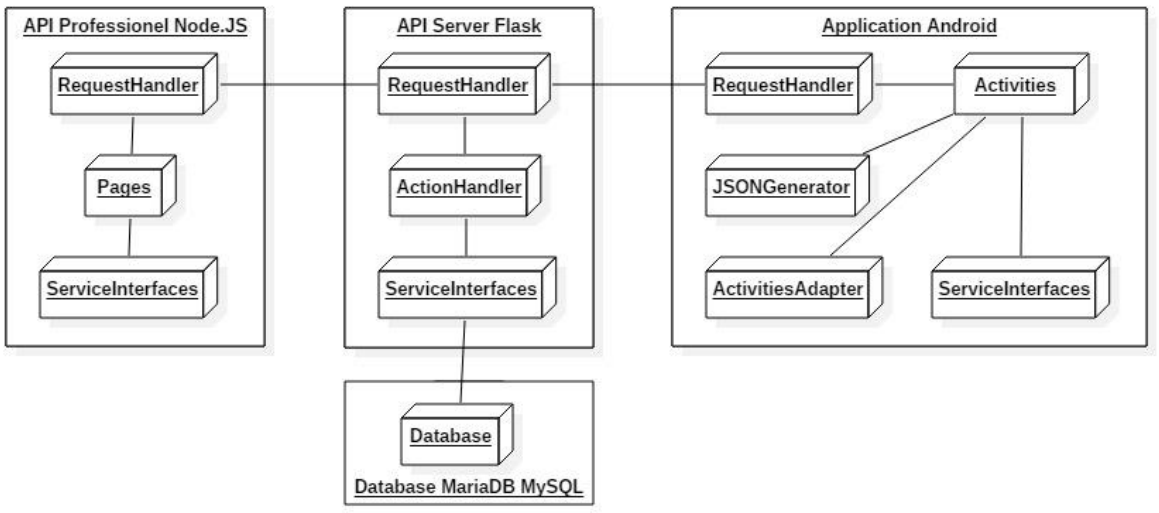
1) Utilisé pour générer une suite de mini-trajets couvrant un départ et une arrivée.
2) Utilisé pour calculer le nombre de points correspondant à une annonce.
3) Utilisé pour l'estimation de volume lorsqu'un utilisateur prend une photo de ses affaires.
4) Utilisé dans le cas où il y a un problème avec le transporteur.
5) Utilisé dans le cas où il y a un problème avec le client.
6) Simple liaison pour la persistance du modèle.
7) Simple liaison pour la persistance du modèle.
8) Simple liaison pour la persistance du modèle.
9) Simple liaison pour la persistance du modèle.
10) Le service de persistance est lui-même connecté à une base de données.
11) Utilisé dans le cas où le transporteur à un problème avec son véhicule.
12) Simple liaison pour la persistance du modèle.
13) Liaison Service de monitoring / Service de persistance.

Schéma des composants à implémenter

Premier cycle de developpement



Second cycle de developpement



Premier cycle de developpement

Nous avons choisi la technologie Android pour la partie application mobile car celle-ci représente la grande majorité des terminaux mobile à travers le monde, de plus, aucun des membres de l'équipe n'est à l'aise avec iOS qui est le second choix possible. De plus, il est beaucoup plus simple, à terme de déployer une application Android au grand public qu'une application iOS sur les stores respectifs.

Pour la partie du service de persistance, nous avons choisi de l'implémenter à l'aide de Symfony 4, un framework PHP. En effet, ce framework offre beaucoup de possibilités tant pour la création d'applications web (qui ne nous concerne pas) que la réalisation de end-point à l'aide d'un système de routes intuitif, d'un système de vérifications de droits ou encore d'un ORM intégré (Doctrine 2). Un des membres de l'équipe est familier avec cette technologie, ce qui nous fera gagner du temps.

Pour la Base de données, nous nous sommes orientés vers une base MySQL. Simple à mettre en place et à déployer, elle reste le choix idéal pour lancer son produit tant que l'on ne se heurte pas à des contraintes de persistances exotiques. De nombreux géants du numérique ont commencé avec ce SGBD (comme Facebook).

La communication entre l'application mobile et la couche de persistance s'effectuera à l'aide de routes et de verbes HTTP.

Second cycle de developpement

La première décision prise dans ce thème est de remplacer le service de persistance en PHP ainsi que son framework Symphony car celui-ci nécessite l'installation de beaucoup d'outils ainsi que d'une manipulation fastidieuse afin de pouvoir le démarrer. De plus, il y a qu'un seul membre de notre équipe de développement qui est suffisamment à l'aise avec cette technologie pour l'utiliser efficacement.

On a exploré plusieurs technologies pour remplacer ce service, comme JavaScript, J2EE ou encore Python. Notre choix s'est donc porté sur le framework Flask de Python.

Le choix de base de Python s'est porté sur la flexibilité de celui-ci dû à ses nombreuses librairies qui nous sera utile dans le cas d'accès à une base de données. De plus, Python nécessite que peu de ressources au démarrage de celui-ci. Pour le framework, on a comparé Flask avec Django, le constat étant que Flask est bien plus utile pour faire une API non graphique grâce à son système de routes.

Pour l'API professionnel à développer, on a rapidement songé à une technologie Web afin de le rendre accessible depuis le plus grand nombre d'appareil. Qu'il soit un terminal mobile ou bien un ordinateur basique.

Pour la technologie Web, on a besoin d'une technologie rapide à prendre en main avec des possibilités pour faire une interface graphique permettant une utilisation aisée pour le transporteur. C'est dans cette optique que nous avons décidé de choisir Javascript qui lie les trois points ci-dessus.

Roadmap

PREMIER CYCLE DE DEVELOPPEMENT

Semaine 42	Semaine 43	Semaine 44
Création des différentes classes à utilisées		
Création d'une IHM simple pour l'application		
Mise en place de la base de données		
Création de l'application Android sans persistance		
	Mise en place de la couche de persistance	
	Ajout des interactions entre l'application Android et la couche de persistance	

SECOND CYCLE DE DEVELOPPEMENT

MERCREDI 16 JANVIER 2019	- Interprétation des nouveaux besoins du client - Création d'une nouvelle User Story - Choix des technologies à utiliser - Mise à jour du diagramme de composants - Créations des architectures des composants à implémenter - Création d'une IHM pour le nouveau module - Création de la nouvelle roadmap - Mise à jour du document d'architecture - Migration du service de persistance en PHP vers une API côté serveur implémentée en Flask - Terminer les routes de l'API côté serveur - Terminer les routes de l'application mobile pour les particuliers
MERCREDI 23 JANVIER 2019	- Modification des algorithmes de tri des annonces côté serveur - Ajout de messages d'erreur dans l'application mobile - Ajout de logs API côté serveur - Modification du document d'architecture - Progression de l'application - Modification des Entities en Service Interfaces
MERCREDI 30 JANVIER 2019	- Création partielle de l'API pour les professionnels en requêtes Postman - Progression de l'application - Création d'une API rudimentaire en Javascript
MERCREDI 06 FEVRIER 2019	- Hébergement serveur local de l'API pro JS - Amélioration de l'inscription de l'API pro JS - La connnection dans l'API pro JS vérifie que le compte soit professionnel - Création de la route de mise à jour du solde dans l'API serveur en Python - Mise à jour du solde de l'utilisateur dans l'API pro JS via la route HTTP - Mise en place d'une démo fonctionnel mais améliorable - Tentative de mise en place d'un système de cohérence sur le serveur Python (via des queues)
MERCREDI 13 FEVRIER 2019	- Cohérence des données entre les différentes vues - Téléchargement d'un csv prérempli avec les numéros d'annonces sélectionnés dans l'API pro en JS - Amélioration de la démo existante

IHM

Premier cycle de developpement

Une IHM simple de l'application a été réalisée afin de mieux concevoir les différentes étapes à implémenter lors de la création de notre application. Celle-ci a été conçu sous Marvel App et déroule entièrement la user story du transporteur particulier.

Le lien de l'IHM : https://marvelapp.com/f2cc2h4/screen/49045251

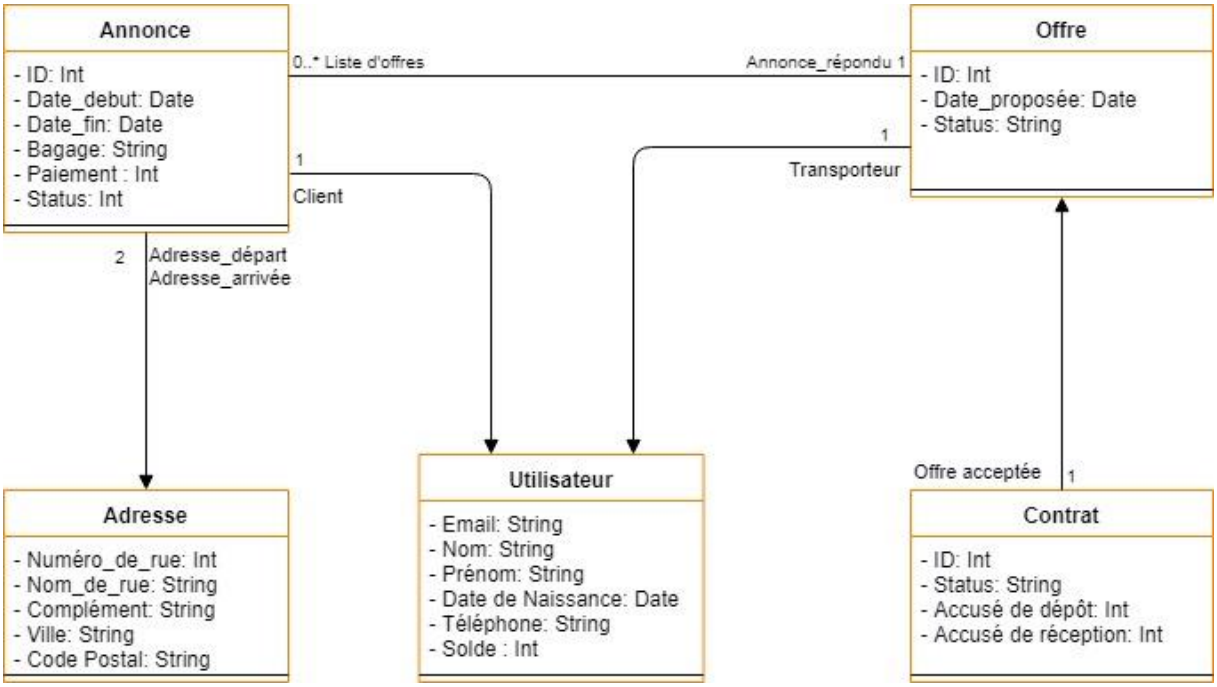
Second cycle de developpement

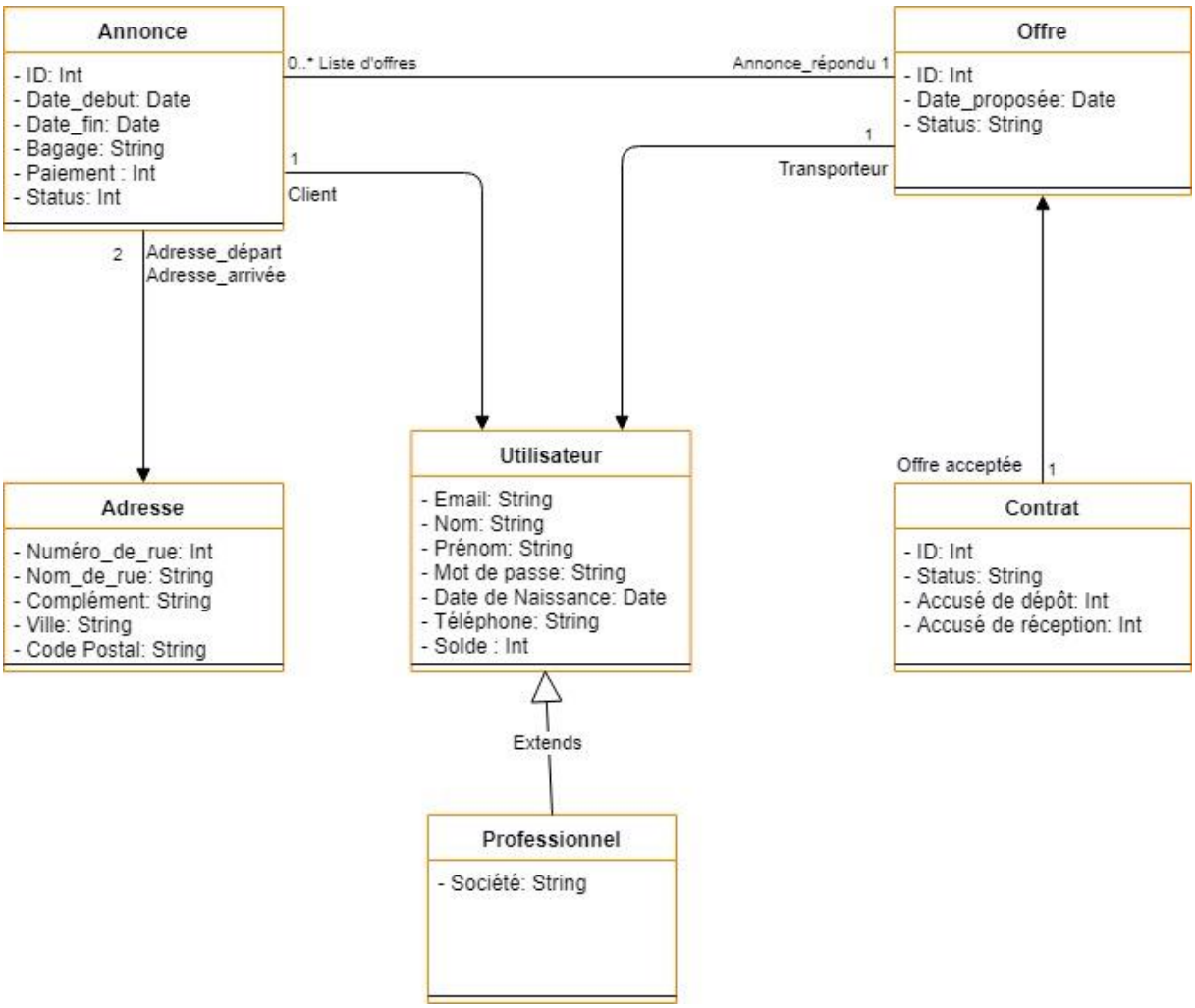
Une IHM a aussi été créer pour l'API professionnel dans le même cadre que l'application mobile, c'est-à-dire une meilleure visualisation du travail à accomplir.

Le lien de l'IHM : https://marvelapp.com/ag2ab01/screen/52327188

Diagramme de classes

PREMIER CYCLE DE DEVELOPPEMENT





La classe Professionnel est une extension de la classe utilisateur (qui représente un utilisateur particulier) en lui ajoutant de manière obligatoire un non de société. On estime cette modification suffisante pour correctement identifier les deux types de transporteurs.