Architecture Logicielle

# Variante 9 : Mobile App for Transporters

## Groupe I

[User Story 2](#_Toc527490556)

[Diagramme de composants global 3](#_Toc527490557)

[Choix des technologies 4](#_Toc527490558)

[Roadmap 4](#_Toc527490559)

[IHM 5](#_Toc527490560)

[Diagramme de classes 5](#_Toc527490561)

User Story

### Premier cycle de développement

|  |  |
| --- | --- |
| **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 développement

Dans le cas du premier cycle de développement, Bob devient un transporteur particulier.

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.

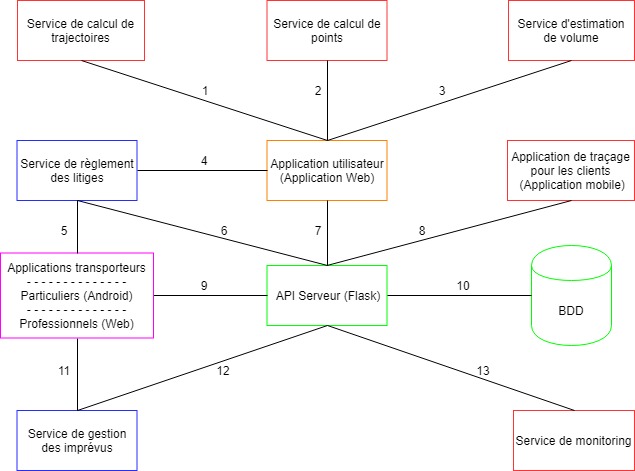
*1/ Transporteurs professionnels avec des camions, prennent plusieurs colis.*

* 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).*

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

Diagramme de composants global



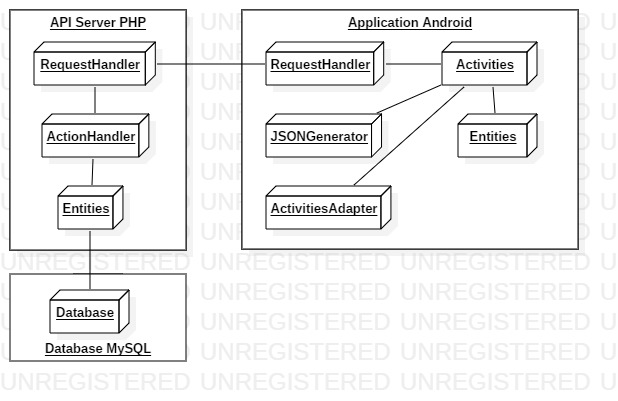
**Légende :**

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 est le seul qui va changer durant le second cycle de développement.

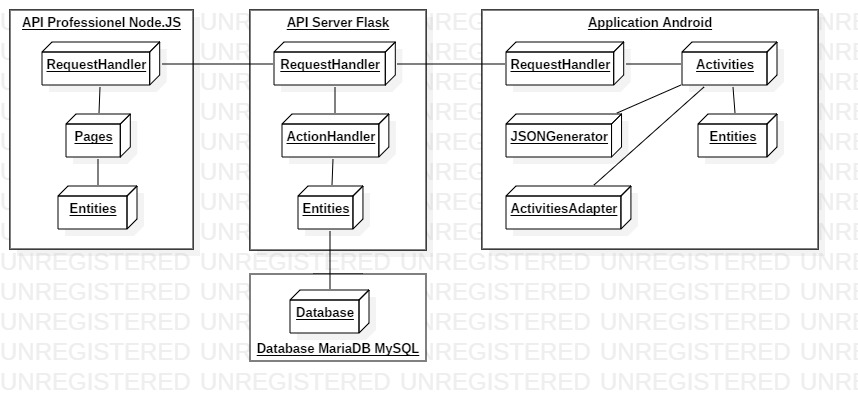
1. Utilisé pour générer une suite de mini-trajets couvrant un départ et une arrivée.
2. Utilisé pour calculer le nombre de points correspondant à une annonce.
3. Utilisé pour l’estimation de volume lorsqu’un utilisateur prend une photo de ses affaires.
4. Utilisé dans le cas où il y a un problème avec 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 développement



### Second cycle de développement



Choix des technologies

### Premier cycle de développement

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 développement

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 Node.JS qui lie les trois points ci-dessus.

Roadmap

### Premier cycle de développement



### Second cycle de développement

|  |  |
| --- | --- |
| 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 * Création d’un algorithme permettant de limiter la taille du panier en fonction de la place disponible dans le véhicule * Ajout de messages d’erreur dans l’application mobile * Création de tests unitaire pour l’application mobile * Ajout d’un outil d’intégration continue |
| Mercredi 30 Janvier 2019 | * Modification de certaines Entities pour permettre l’implémentation de la nouvelle API * Création de l’API pour les professionnels en requêtes Postman * Modification de l’API serveur pour la rendre compatible avec l’API pro * Modification de l’application mobile pour la rendre compatible et cohérente avec l’API pro |
| Mercredi 06 Février 2019 | * Migration de l’API pro en requête Postman vers une API web Node.JS |
| Mercredi 13 fevrier 2019 | * Raffinement de l’API Node.JS |

IHM

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.

Le lien de l’IHM : <https://marvelapp.com/f2cc2h4/screen/49045251>

Diagramme de classes

