Integration de services
2017 – 2018

# Virtual Travel Agency


Étudiants :

- ABDELMOUGHIT AFAILAL
- GUILLAUME FAGUET
- MAXIME DEJEUX
- ILIAS NAAMANE

Encadrants

- M. Sébastien Mosser
- Mme Mirelle Blay-Fornarino


Université Nice Sophia Antipolis

POLYTECH NICE-SOPHIA

*Résumé :*

Ce Lab est dans le cadre d'un projet de la matière "Intégration de Services" et qui consiste à intégrer plusieurs services et de les faire communiquer via un bus.
Pour ce faire on a utilisé l'outil open source Apache servicemix qui est un bus de service d'entreprise distribuée basé sur le modèle d'architecture orienté service.

# 1. Architecture globale

L'architecture des flows d'intégrations est composée de deux parties :

- Une partie "**Business Travel"** qui traite la recherche d'un vol/voiture/hôtel pour un voyage et envoie le ticket composé de ces 3 éléments au service "BusinessTravel".

- Une partie "**Remboursement**" qui traite les évidences (photo des factures) envoyées par mail par l'employé pendant son voyage pour qu'il se fasse rembourser.

Nous avons décidé de faire deux flows d'intégrations car le voyage de l'employé est divisé en deux parties. Le flow "Business Travel" est utilisé avant le départ de l'employé tandis que le flow "Remboursement" s'effectue pendant(et/ou après) le voyage de l'employé.

# 2. Flow d'intégration

## Flow Business Travel:

Le flow Business Travel dispose de 6 Services qui communiquent entre eux afin de fournir le meilleur résultat en terme de coût de voyage et puis envoyer la requête pour le 7éme Service Business Travel.

- 2 Services Vols
- 2 Services Hotels
- 2 Services Voitures
- 1 Service Business Travel

Durant le Lab 1 on a pu concevoir et développer 4 services ( Vol, Hotel, Voiture et Business Travel ).

Durant la phase d'intégration on a pu choisir 1 service par métier ( Vol,Hotel,Voiture) des services des autres groupes et qui sera le plus adaptable et facile à intégrer.

- **Le service Car** que nous avons choisi d'intégrer, est le service du groupe 9 et qui est un service REST bien conçu exposant une ressource http://host/cars permettant de trouver toutes les voitures disponibles selon une destination et une date start, date end et accepte le tri par ordre croissant et cela nous facilitera le fait de retrouver la voiture de location la moins chère pour le flow d'intégration.

- **Le service Vol** que nous avons choisi est un service du groupe 6 et qui est un service document similaire à notre paradigme.

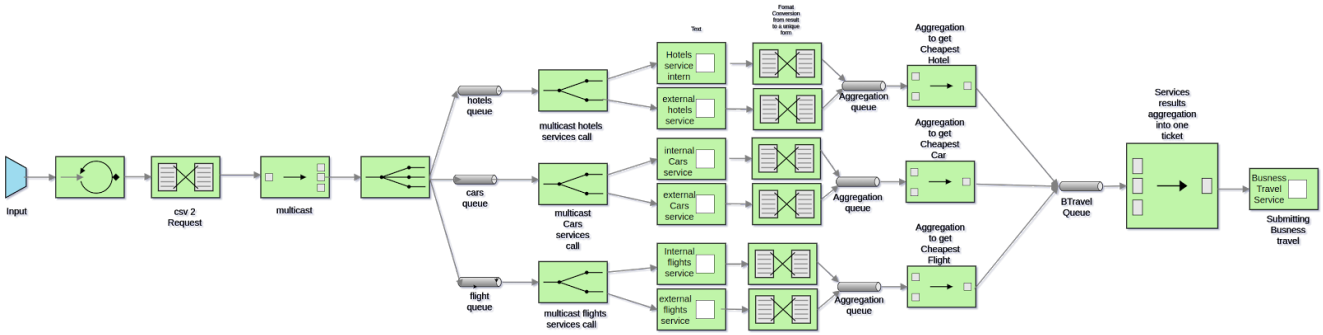
- **Le service Hotel** est le service du groupe 1 et qui est un service REST facile à intégrer vu qu'il ne se soucie pas des stubs, WSDL comme il est le cas pour le RPC, et le résultat retourné par ce service correspond très bien à notre service interne "hotel".

Les services dont on dispose en interne:

- **Service Car** en RPC
- **Service Hotel** en RPC
- **Service Vol** en Document
- **Service Business Travel** en REST

## Architecture du flow Business Travel:

Le principe de cette intégration, est de faire communiquer les 2 services cars entre eux, les 2 services hôtels entre eux, les 2 services vols entre eux, afin d'avoir l'hotel le moins cher, le vol le moins cher, la voiture la moins chère et par la suite envoyer la requête au Business Travel. Pour effectuer ceci on a implémenter le flow d'intégration suivant:




Notre architecture dispose d'un fichier CSV en input facile à parser contenant les spécifications de l'employé ( destination, date début, date sortie) et à transformer en hashMap et qui contient la destination, date début et date fin. Le CSV est ensuite transformé en un objet sérialisé requête qui se multicast et traverse 3 queues ( car, hotel,flight ) et pour chaque queue on se retrouve avec un autre multicast afin de pouvoir faire des calls à chacun des services ( par exemple flight 1, flight 2 ) afin de retourner l'élément le moins cher pour chaque service. Par la suite, on effectue une agrégation entre les différents services similaires après avoir standardiser leurs réponses en un seul et unique format (objet), afin de pouvoir comparer les deux résultats et retourner l'élément le moins chère des 2 services identiques en format Json. Finalement on concatène lors d'une aggregation les 3 réponses (car,hotel, flight) pour ainsi former la requête business travel qui est en REST et ensuite avoir par la suite un output.txt qui affiche le message "**business travel sumbitted successfully and waiting for approval".**

Le choix de format de donnée JSON comme sortie de chaque métier est primordial, vu qu'on dispose d'un service Business Travel en REST et qui accepte que du JSON. Donc ce serait facile pour l'agrégation de concaténer les 3 réponses pour ainsi former la requête au business travel.

## Gestion des exceptions pour le business travel:

Dans une architecture microservices, la gestion de panne des services est très importante. Dans notre cas on dispose d'un script.sh qui permet de stopper aléatoirement les differents services.

Lors de la phase de développement on a pu gérer les exceptions cas par cas:

- 1 seul service d'un métier

  Dans ce cas, automatique la réponse qui sera prise en compte est la réponse du métier disponible. Et pour gérer ceci, après avoir capturer l'exception on crée un Fake élément avec un prix= Integer.max_value pour que quand il soit comparé avec le résultat de l'autre service, c'est le service disponible qui sera considéré.

- 2 services du même métier

  Dans ce cas, aucune réponse ne sera transmise au business travel concernant ce métier.

## Flow Remboursement:

Pour le création du flow remboursement nous avons créé deux services :

- Le service OCR : le service prend en entrée un JSON. Il récupère le nom de l'image dans celui-ci est lui ajoute un montant prédéfinis en fonction du nom de l'image, puis retourne le JSON.
- Le service service_report : Le but de ce service est gérer une base des données contenant des rapports appartenants aux employés. la raison de cette architecture est de respecter le fait que le manager peut accéder à un rapport même si ce rapport n'est encore terminé. ce service permet d'ajouter , modifier,supprimer,lister,soumettre un rapport. Quand l'utilisateur décide de mettre fin au rapport et il le soumettre, le

service vérifie si le rapport est valide, c-à-d, est ce que le remboursement est accepté automatiquement, ou bien non, ensuite il met à jour la base des données contenant tous les rapports. Ce service suit le paradigme document, il consomme et produit un résultat en format Json et il traite les events suivantes ( Append, List,Create, ListByRptID, ListByEmPID, submit").

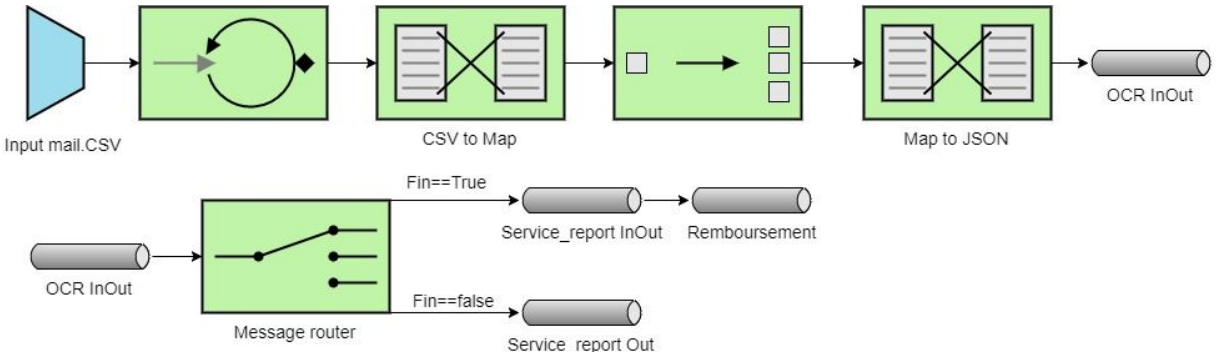
**-Append** : prend l'id employé , id busnessTRravel , durée du business travel , et le montant, et permet d'ajouter la dépense à un rapport existant au nom de l'employé sinon il crée un nouveau.s'il le rapport est complet et valide il renvoie le rapport en réponse, sinon il envoie une indication que le rapport a été bien modifié.

**-List** : permet de lister tout les rapport qui existent sur la base des données, c'est possible de mettre deux préférences,"valide" et "complet" pour lister les rapports validés/non validés ( complets/non complets ).

**-Create** : prend l'id employé , et permet de créer un nouveau rapport et de l'ajouter sur la base des données

**-ListByRptID / ListByEmPID :** permettent de retourner un rapport à partir de l'id rapport ( id employé ).

## Architecture du flow




## Déroulement du flow

Le point d'entré est un dossier qui simule une boite mail. On y ajoute un fichier "mail.csv" qui peut contenir les valeurs d'un ou plusieurs mail. On commence par transformer le fichier CVS en une liste d'objet (Map), puis nous récupérons chaque objet contenu dans la liste pour les transformer en JSON (ces traitements son effectué en parallèle).

Ces JSON sont ensuite envoyé à l'OCR qui nous permet de récupérer le montant de chaque "image".
La valeur "fin" du JSON est ensuite analysé afin de savoir si c'est le dernier mail que va envoyer l'employé.
Si c'est le cas alors on envoi le JSON au service "Service_report" qui va ajouter les informations contenu dans le JSON dans le rapport final et va le retourner sous forme de JSON. Nous enverront ensuite ce JSON dans une queue "Remboursement"(bouchon) qui représentera l'envoi à un vrai service de remboursement.
Si "fin" est égale à false alors nous envoyons le JSON au service "Service_report" pour qu'il ajoute les informations au rapport de l'employé.
La gestion des seuils est géré à l'intérieur du service "Service_report" et nous considérons que le manager à un accès à ce service pour regarder les justification et pour éventuellement les approuvés.

# Technologies

Pour ce projet nous avons utilisé les technologie suivantes :

- MongoDB pour le stockage des informations dans les services "BusinessTravel" et "Vol" en document.
- Les services ont été codés avec le langage JAVA sauf les services "BusinessTravel" et "Car externe" qui sont en JavaScript.
- Apache servicemix en tant que bus.
- Le framework Apache Camel pour le routage des messages entre les applications et le Java avec le DSL pour la configuration du composant.

# Conclusion

## Fonctionnalités terminées

- Le flot du business travel.
- Le deuxième Flot remboursement

## Fonctionnalités à terminer:

- Gestion des exceptions
- Tests unitaires
- Faire un main services
- Corriger le bug présent dans le flow remboursement pour router les messages

## Guide d'utilisation

1. ./build.sh
2. ./start.sh
3. cd deployment ./run-bus.sh
4. Dans servicemix : cd bin/ puis ./client
5. Si ./client ne fonctionne pas il faut alors exécuter cette commande avant : ./karaf

## Répartition des tâches :

Tous les membres du groupe ont contribué au maximum à ce projet, donc on répartit la note avec une part équitable de 25% chacun.