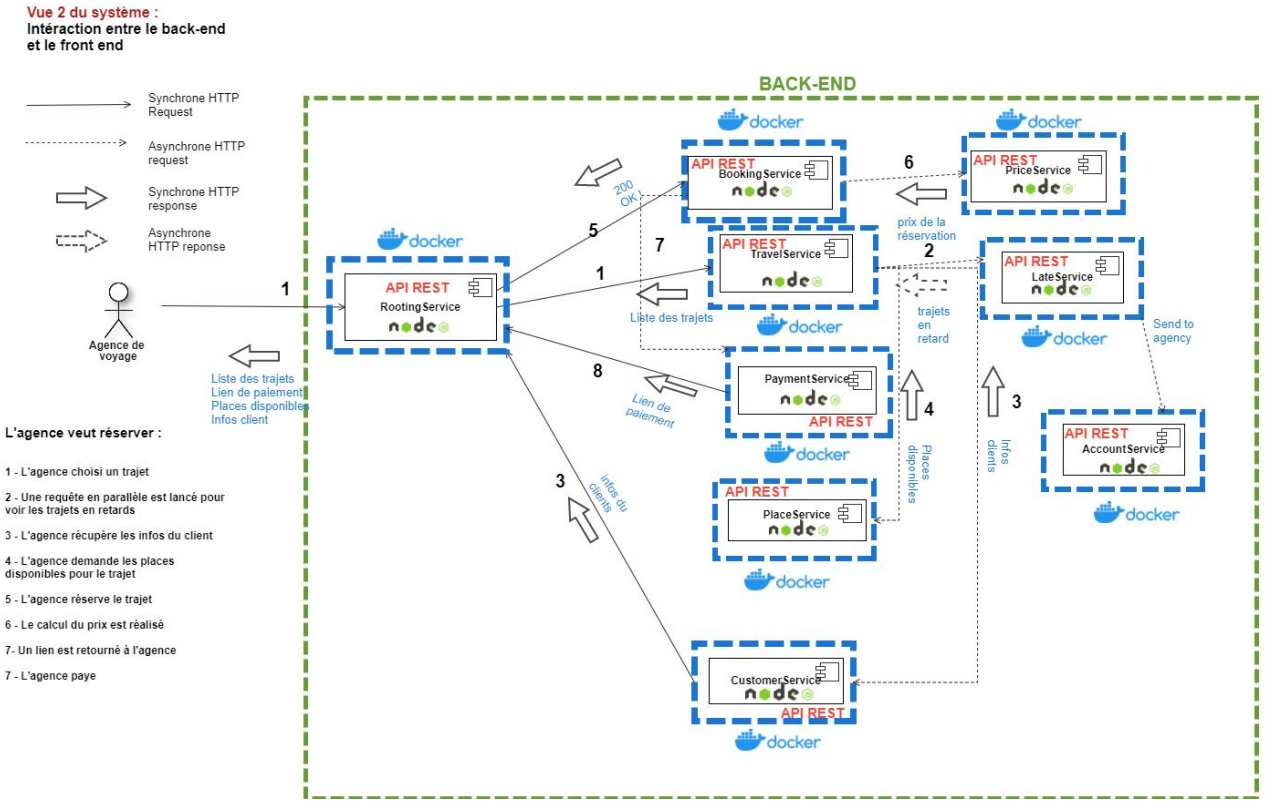
AL Si5-team-h

Maël DELABY
Fabrice SIMON
Yassine JRAD
Adrien VASSEUR

Ancienne architecture



Nouvelle demande du client

- Vente de l'exploitation des trains à PLMCF
- 2 nouvelles instances du système de réservation

Qu'est-ce que cela implique ?

- Répartition des trains, et donc des trajets entre les deux compagnies, mais aussi des données clients,etc...

Challenge

- L'agent qui réserve via l'API ne doit pas voir de changements de son côté, et doit pouvoir réserver sans savoir où sont les données des deux compagnies

Nouveaux changements à faire ?

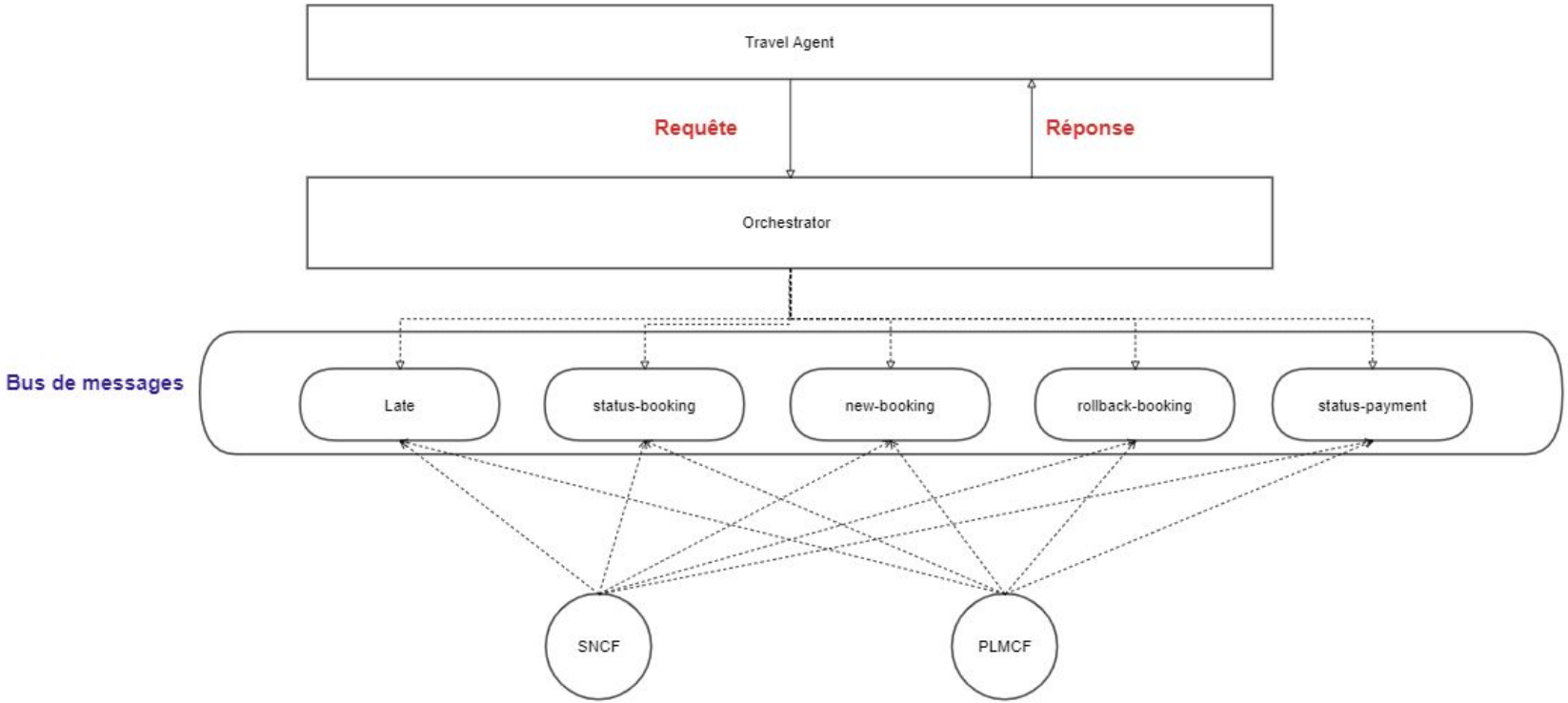
Avant

- Un seul système de réservation
- Suivi de la réservation simple à maîtriser, gestion interne

Après

- Plusieurs systèmes de réservation
- Plusieurs suivis
- De multiples transactions à gérer… si une transaction échoue dans un des systèmes, comment faire remonter l'information ? Comment maintenir un tel système ? Si d'autres ventes se font alors + de système de réservations

Evolution de l'architecture



Technologies utilisées pour le changement

Pourquoi ?

- Tolérant à la panne
- Scalable
- Faible latence
- Haut débit (+ de milliers de messages /sec)

- Notre système devra gérer des milliers de transactions chaque jour, l'information doit donc être remonté le plus rapidement possible, et perdre l'avancé d'une transaction n'est pas tolérable

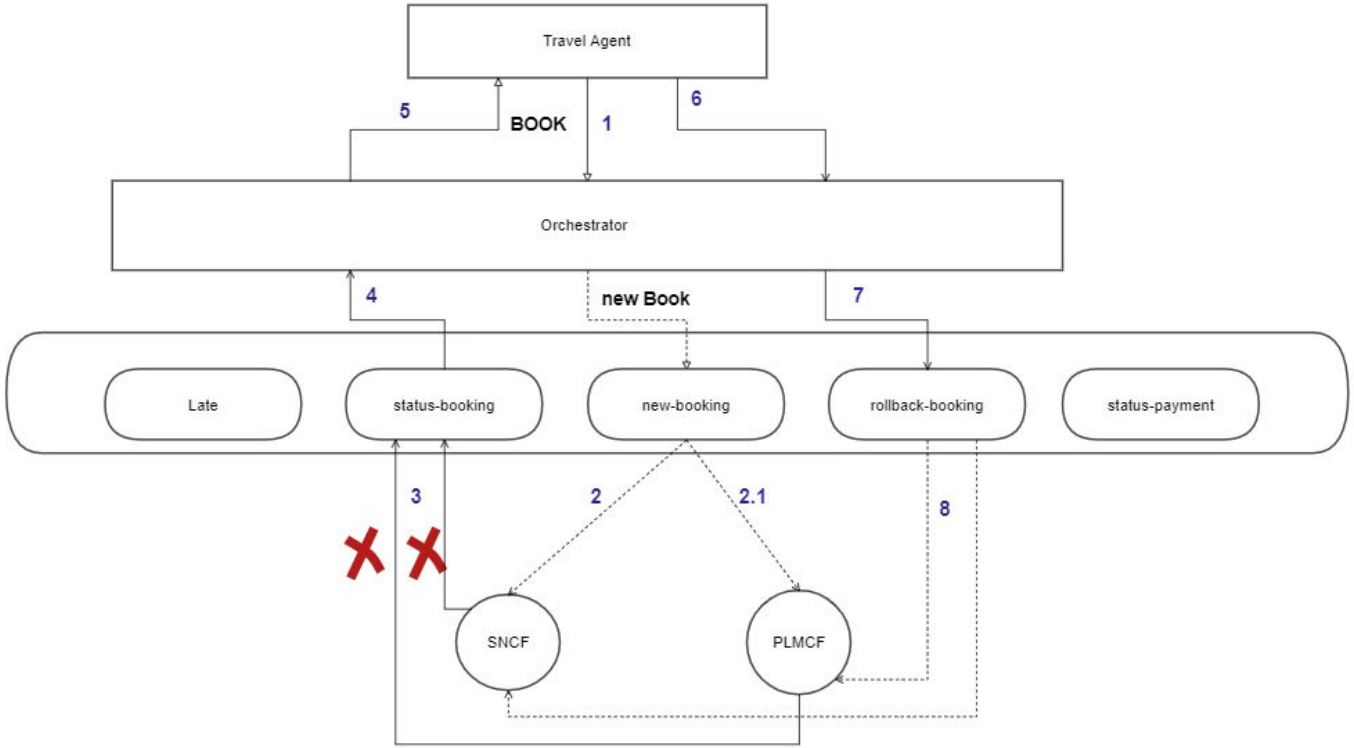
Nouveau scénario

Contexte : l'agence fait une réservation pour Nice comme départ et Brest comme destination, le seul trajet possible est un trajet avec correspondance

Scénario :

- l'agent réserve les trajets avec les options du client (emplacement vélo et prise de charge)
- Le système fait la réservation
- L'agent reçoit un message d'erreur indiquant que la réservation a échoué...
- Le système annule la réservation

Scénario


Travel Agent
5
BOOK
1
6
Orchestrator
4
new Book
7
Late
status-booking
new-booking
rollback-booking
status-payment
3
2
2.1
8
SNCF
PLMCF

Démo