

Team H

Scope du projet :

Fonctionnalités gérées par l'API :

- La création d'un compte "agence de voyage"
- Afficher un voyage et calculer son prix selon les informations des clients (Âge, Carte(s) réduction) et de la réservation (première classe / deuxième classe, départ / destination, date et heure, options (prises électriques, vélo, mobilité réduite, wifi, assurance annulation))
- Réservation d'un voyage
- Visualiser des sièges précis dans un train (savoir combien de personnes maximums peuvent être à côté)
- Annuler une réservation
- Remboursement d'une réservation si assurance
- Modification d'une réservation
- Informer le client si leur train est en retard avec un mail
- Payer une réservation

Fonctionnalités non gérées :

- Problème extérieur causant une annulation d'un départ de train
- Création de trajet, train, destination...
- Promotions

Personas / Users :

- Logiciel ou application utilisé par l'agence de voyage
- Service de paiement externe

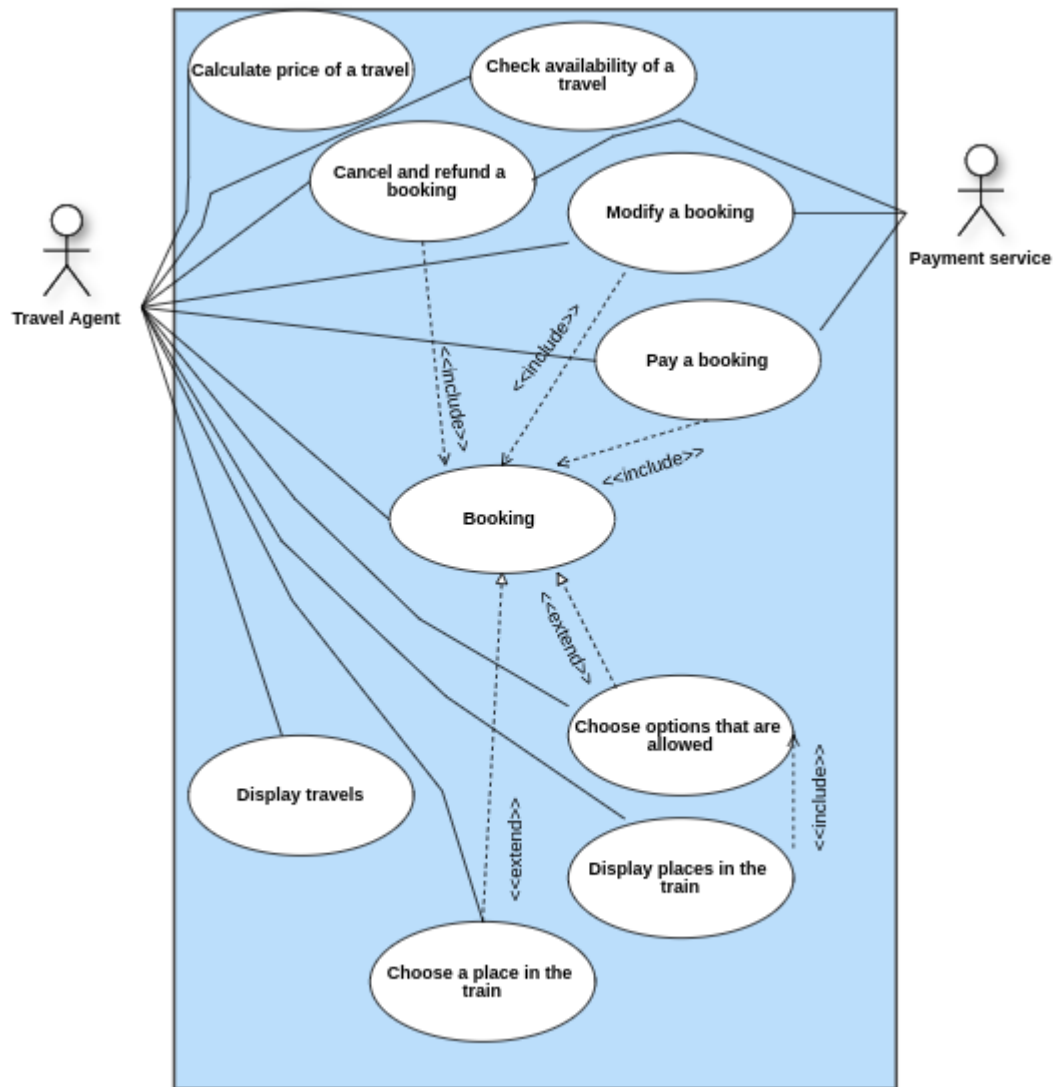


Diagramme de cas d'utilisation :

Les principaux scénarios :

1) Une agence souhaite pouvoir utiliser l'API

- 1 - L'agence envoie une requête contenant diverses informations (email, nom...)
- 2 - Le système renvoie un identifiant unique que l'agence utilisera pour toutes ses prochaines requêtes

2) Une agence souhaite réserver un voyage pour un client

- 1 - L'agence envoie une requête au système contenant les informations d'un homme âgé de 25 ans voulant un trajet Nice - Bordeaux à la date du 6 novembre 2020 à 9h en deuxième classe, avec aucune option
- 2 - Le système renvoie une liste à l'agence contenant les différents trajets possibles
- 3 - L'agence réserve un trajet
- 4 - Le système envoie le prix de la réservation au service de paiement
- 5 - Le service de paiement retourne un lien de paiement au système
- 6 - Le système renvoie ce lien à l'agence
- 7 - L'agence procède au paiement du billet via le lien
- 8 - Le service de paiement confirme le paiement au système
- 9 - Le système envoie le billet à l'agence

Variantes de scénarios :

2.a reservation pour un groupe de clients hétéroclites

- L'agence envoie une requête à l'API contenant un groupe d'une famille (un adulte de 20 ans avec une carte de réduction jeune , une femme de 40 ans avec un vélo et un homme de 65 ans avec mobilité réduite) souhaitant réserver
- Le système renvoie une liste à l'agence contenant les différents trajets possibles avec des prix étant l'addition des différents prix pour chaque personne dans le groupe

- L'agence envoie le numéro du trajet et le numéro de train au système
- Le système renvoie les emplacements pour vélos disponibles
- L'agence de voyage choisi une place et l'envoie au système
- Le système confirme le choix de l'emplacement
- L'agence fait une réservation du trajet pour chaque personne dans le groupe
- retour étape 4

2.b choisir les places avant la réservation

- L'agence envoie le numéro du trajet et le numéro de train au système
- Le système renvoie les places disponibles dans le train
- L'agence de voyage choisi une place et l'envoie au système
- Le système confirme le choix de la place
- retour étape 3

2.c retard d'un train

- Le système envoie la liste des trajets disponibles et des trajets en retard
- retour étape 3

2.d choisir un emplacement de vélo

- L'agence envoie le numéro du trajet et le numéro de train au système
- Le système renvoie les emplacements pour vélos disponibles
- L'agence de voyage choisi une place et l'envoie au système
- Le système confirme le choix de l'emplacement
- retour étape 3

3) Une agence souhaite annuler une réservation avec remboursement

- 1 - L'agence envoie une requête au système afin d'annuler sa réservation
- 2 - Le système envoie une requête au service de paiement pour lui notifier du remboursement
- 3 - Le service de paiement envoie un lien au système
- 4 - Le système renvoie ce lien à l'agence
- 5 - Une fois le remboursement effectué le service de paiement prévient le système
- 6 - Le système envoie un mail à l'agence pour confirmer l'annulation

Variantes de scénarios :

3.a (Sans remboursement)

- Suppression des tâches 2, 3, 4 et 5

4) Une agence souhaite modifier une réservation

- 1 - L'agence envoie une requête avec le numéro de la réservation qu'elle souhaite modifier
- 2 - Le système lui répond en lui disant quels champs modifier
- 3 - L'agence envoie une requête au système pour demander les trajets disponibles avec des informations dans les champs différents
- 4 - Le système lui répond les trajets disponibles pour les nouvelles informations
- 5 - L'agence envoie une requête au système avec les modifications apportées
- 6 - Le système envoie un mail avec le nouveau trajet

Variantes de scénarios :

4.a (procéder à un paiement pour la modification) :

- Le service de paiement retourne un lien de paiement au système

- Le système renvoie ce lien à l'agence
- L'agence procède au paiement du billet via le lien
- Le service de paiement confirme le paiement au système
- Le système lui répond les trajets disponibles pour les nouvelles informations

5) Afficher les différentes réservations existantes :

- 1 - L'agence demande les réservations existantes
- 2 - Le système renvoie la liste des réservations existantes

Variantes :

- 5.a - L'agence envoie plusieurs informations en plus du numéro (information client et/ou trajet) concernant les réservations
- 5.b – Le système ne renvoie aucune réservation dans la liste

Scénario pour le POC de Novembre :

Pour le POC du 6 Novembre, nous allons réaliser le scénario 2: “Une agence souhaite réserver un voyage pour un client”

Road Map

WEEK_41 : component_diagram & road map

Release 1 (6 nov) :

MVP avec dedans :

Scénario 2 + variantes (réservation avec plusieurs clients)

Release 2 :

Scénario 1 (clef d'API)

Scénario 5 (liste des réservations)

Scénario 3.a (annulation sans remboursement)

Scénario 4 (modification réservation)

Release 3 :

Scénario 5.a (liste des réservations plus poussé)

Scénario 3 (annulation avec remboursement)

Scénario 4.a (modification réservation avec surplus ou sous plus)

RoadMap de la MVP (jusqu'au 6 novembre) :

du 09/10 au 16/10 :

Nouvelles fonctionnalités :

- L'agence peut choisir un trajet parmi les trajets proposés
- L'agence peut réserver un trajet pour un client sans aucune options ni choix de place
- L'agence peut utiliser le service de paiement pour payer

Ajout de 4 services :

RootingService -> Gateway pour appeler les services (réduit le couplage entre les services)
TravelService -> renvoie tous les trajets (un seul train) sans prise en compte du profil client
BookingService -> Créer une réservation simple sans prise en compte d'options ni de place
PaymentService -> Permet de renvoyer un lien de paiement pour un client

du 16/10 au 23/10 :

Nouvelles Fonctionnalités :

- L'agence peut choisir des options pour un client lors de la réservation
- L'agence peut voir et choisir des places dans un train
- L'agence peut choisir un trajet selon le profil du client donné

Ajout de 2 services :

"PlaceService" -> Possibilité de choisir une place, de voir les places disponibles
"PriceService" -> Possibilité de calculer le prix selon les options et profil du client

Amélioration de services :

TravelService -> renvoie tout les trajets (un seul train) avec prise en compte du profil client
BookingService -> Créer une réservation avec prise en compte options et profil client

du 23/10 au 30/10 :

Nouvelles Fonctionnalités :

- L'agence peut choisir des options lors de la réservation
- L'agence peut réserver pour un groupe de personnes avec options et choix de places possibles

Amélioration de service :

TravelService -> renvoie tous les trajets (plusieurs trains) avec prise en compte d'un groupe de client
BookingService -> Créer une réservation avec prise en compte options et différents profils clients (groupe)

du 30/10 au 5/11 :

Nouvelles Fonctionnalités :

- L'agence peut voir les trajets en retard
- L'agence peut récupérer les informations des clients
- L'agence peut créer un compte en son nom pour enregistrer toutes ses réservations

Ajout de services :

LateService -> permet d'informer les différents retards d'un trajet
CustomerService -> permet de récupérer les infos clients
AccountService -> permet de créer un compte pour l'agence

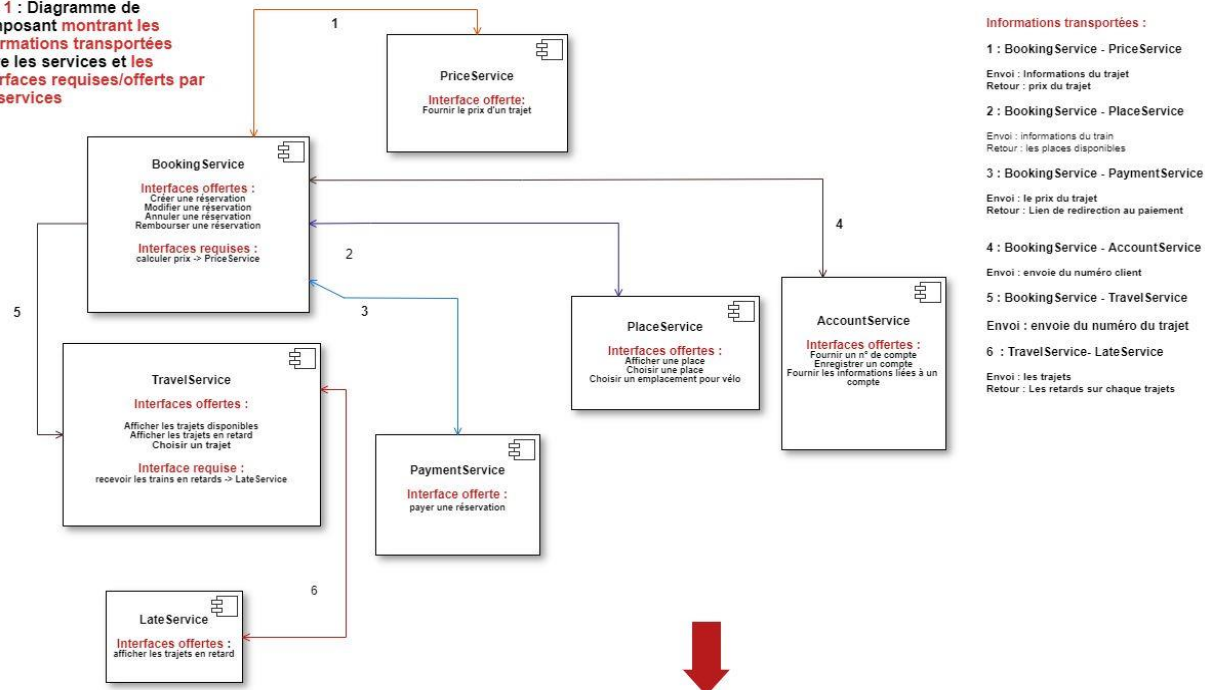
Améliorations de services :

-> TravelService -> renvoie les trajets ayant un retard

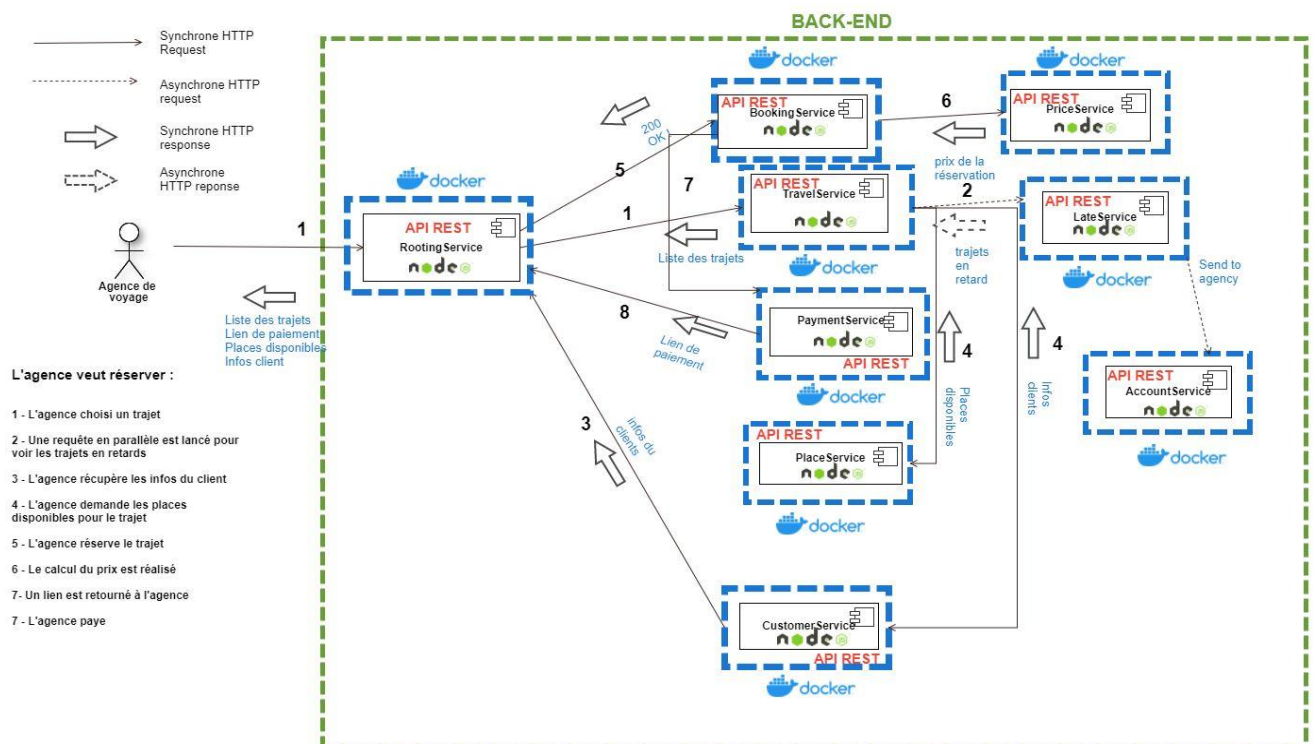
Diagrammes composants et interactions front-back end

[Voir le diagramme](#)

Vue 1 : Diagramme de composant montrant les informations transportées entre les services et les interfaces requises/offertes par les services



Vue 2 du système : Interaction entre le back-end et le front end



PriceService :

Service de calcul des prix des trajets en fonction des options sélectionnés, des catégories d'âge des clients, distance train etc.

BookingService :

- Créer une réservation
- Modifier une réservation
- Annuler une réservation
- Remboursement d'une réservation

PlaceService :

- Visualiser les places dans le train
- Choisir une place dans le train
- Choisir un emplacement pour vélo

TravelService :

- Afficher les trajets disponibles pour différentes destinations
- Afficher les trajets en retards

AccountService :

- Enregistrer un compte
- Fournir un numéro de compte
- Créer un compte personnel pour l'agence

PaymentService :

- Payer une réservation
- Se faire rembourser une réservation

LateService :

- Informer d'un retard de train

CustomerService :

- Récupérer les infos des clients

RoutingService :

- Passerelle pour se connecter avec les autres services

Choix technologiques :

Microservices :

Partir sur une application monolithe était possible pour le POC, cependant on a préféré partir sur du microservices pour plusieurs raisons :

-> Libre sur le choix d'une technologie sur chaque microservice

-> Chaque partie de l'application peut être modifiée indépendamment sans devoir modifier toute l'application

Comme notre application va beaucoup évoluer pouvoir être flexible est important, c'est pourquoi on a décidé de partir sur du micro service même si la propriété "Scalabilité" n'est pas pour l'instant intéressante.

API Rest : services simples axées sur les ressources.

- BookingService
- PlaceService
- TravelService
- LateService
- AccountService
- CustomerService

-> Couplage faible client - serveur -> peut être utilisé par d'autres applications mobile/web (pour les autres équipes sur le projet c'est très important)
-> Facilement extensible et portable
-> support native du cache

SOAP : (Pour l'instant REST, dans le futur passé à SOAP)

- PaymentService

-> Transmission de données hautement sécurisée (WS Security)

Node JS :

- Serialisation - deserialisation en json native
- Basé sur une boucle événementielle lui permettant de supporter de fortes montées en charge (requête non bloquantes)
- Communauté grandissante
- Techno stable
- Un seul thread -> léger

PM2 :

- Choix intéressant pour faire du monitoring sur les différents micro-services

-> Permet de gérer les logs venant de tous les micro services
-> Permet de voir les ressources utilisées par chaque microservice et son état de "vie" (online/offline)

Docker :

- Containeurs légers
- Déploiement rapide
- Isolé

Stockage :

- Local JSON

Pour l'instant tout est en local, on a estimé que pour la MVP c'était suffisant cependant plus tard on évoluera vers des BD SQL ou NoSQL en fonction des données des différents microservices

Changements par rapport à l'archi initial :

CustomerService -> oubli d'un service qui contient les infos des clients

PaymentService -> REST, pas de SOAP -> car plus facile à implémenter en Javascript, néanmoins on pense changer plus tard

Faiblesses de notre architecture actuelle :

- Chemin complexe entre les microservices

Solutions envisagées :

- Passer par un bus évènementiel pour réduire le couplage entre micro-services (découplage entre les micro-services)

Auto-évaluation :

	Maël	Fabrice	Adrien	Yassine
Maël	100	100	100	100
Fabrice	100	100	100	100
Adrien	100	100	100	100
Yassine	100	100	100	100