**Document de spécifications techniques du méta-système**

Date de création : **21/10/2013**

Version : **1.6 du 25/04/2014**

SpÉcifications Techniques

MÉta-systÈme

Sommaire

1 Historique des modifications 3

2 Documents applicables 4

3 Introduction 5

4 Les interfaces du système 6

4.1 Les interfaces de recherche d’itinéraire et de localités 6

4.1.1 Protocole 6

4.1.2 Les services de l’interface du méta-système 7

4.1.3 La définition des structures échangées par les services 9

4.2 Le modèle des méta-données 13

4.3 Les interfaces génériques requises sur les SIM 16

4.4 Les interfaces dans l’architecture de déploiement 20

5 Les spécifications techniques générales des systèmes de la solution 22

5.1 Définition du graphe de SIM 22

5.2 Définition de l’itinéraire partiel 23

5.3 Notations 23

5.4 Le système Aiguilleur 24

5.4.1 Introduction 24

5.4.2 Spécification des principales fonctions 24

5.4.3 Phases de développement à venir 34

5.5 Architecture du système d’administration 34

5.5.1 Spécification des principales fonctions 35

5.5.2 Phases de développement à venir 36

5.6 Architecture du site WEB de démonstration 36

5.6.1 Le formulaire de requête 36

5.6.2 L’affichage des résultats de la recherche 36

5.6.3 Technologie d’implémentation 37

5.6.4 Phases de développement à venir 37

6 Conclusion 38

# Historique des modifications

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date de publication | Description des changements | Auteur |
| 1.3 | 24/10/2013 | Création du document | Marc FLORISSON  Stephan SIMART |
| Après diffusion, l’AFIMB a transmis ses remarques listées dans le tableau Revue-AFIMB-V1.3.xlsx | | | |
| 1.4 | 20/12/2013 | Prise en compte des remarques :  3, 4, 12, 14, 20, 23, 28, 32, 36, 37, 41, 42, 43, 45, 46, 49, 53, 54, 55, 56  à faire :  25 (regénérer le schéma de bd)  44 (màj du diag de seq à 2 SIM : rajouter « optimisée » sur l’étape 3)  47 : idem 44 pour le diag de seq à 3 SIM | Marc FLORISSON |
| 1.5 | 25/02/2014 | Rectification du nombre de réponse sur un Appel RI non détaillée n,m  Ajout d’un chapitre qui rassemble les documents applicables  Modification sur la méta-base | Marc FLORISSON |
| Après diffusion, l’AFIMB a transmis ses remarques listées dans le tableau Contribution\_APII\_SIM\_060414.xls | | | |
| 1.6 | 25/04/2014 | Prise en compte des remarques :  1,2,3,7,8,9,11,12,13,14,16,17,19,20,  24,25,27,29,30,32,33,35,36  Cette version du document est associée à la version 1.1 des structures d’échanges (cf document applicables) qui intègrent les remarques :  48,49,50,53,54,55,57,58,61,62,63 | Marc FLORISSON |

# Documents applicables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Titre | Contenu | Date | Version | Auteur |
| XSD-APII-SIM-V1.1.zip | Définitions des structures d’échange de l’interface | 25/04/2014 | 1.1 | Marc FLORISSON  Stephan SIMART |

Cette archive rassemble l’ensemble des définitions des structures échangées

* au niveau de l’interface publique du méta-système
* au niveau de l’interface générique des SIM

Le tableau ci-dessous détaille les fichiers contenus dans l’archive, en indiquant pour chacun l’espace de nommage défini ainsi que les interfaces qui échangent les structures définies :

|  |  |  |  |
| --- | --- | --- | --- |
| Fichier XSD | Interface SIM | Interface méta-système | Espace de nommage |
| Dossier netex | X | X | <http://www.netex.org.uk/netex>  Ce dossier contient un fichier de définition limité aux éléments utilisés |
| Dossier gml | X | X | <http://www.opengis.net/gml/3.2>  Ce dossier contient les fichiers de la version 3.2.1 de GML |
| Itinerary.xsd | X | X | http://www.apiisim.fr/common/1.0/itinerary |
| Itinerary-Request.xsd | X | X | http://www.apiisim.fr/common/1.0/itinerary-request |
| Site.xsd | X | X | http://www.apiisim.fr/common/1.0/site |
| MisLocationTime.xsd | X |  | <http://www.apiisim.fr/mis-generic/1.0/location-time> |
| MisProtocol.xsd | X |  | <http://www.apiisim.fr/mis-generic/1.0/protocol> |
| MisPlanTrip.xsd | X |  | <http://www.apiisim.fr/mis-generic/1.0/plantrip> |
| MisPlanSumedUpTrip.xsd | X |  | <http://www.apiisim.fr/mis-generic/1.0/plantrip> |
| MisCollectStops.xsd | X |  | <http://www.apiisim.fr/mis-generic/1.0/collect-stops> |
| MisCapabilities.xsd | X |  | http://www.apiisim.fr/mis-generic/1.0/capabilities |
| Protocol.xsd |  | X | http://www.apiisim.fr/distributed-journey-planner/1.0/protocol |
| PlanTrip.xsd |  | X | http://www.apiisim.fr/distributed-journey-planner/1.0/plantrip |
| SearchPoints.xsd |  | X | <http://www.apiisim.fr/distributed->journey-planner/1.0/search-point |
| ProtocolFramework.xsd | X | X | http://www.apiisim.fr/common/1.0/protocol-framework |

# Introduction

Ce document fait suite aux documents « état de l’art », « besoins et contraintes technique » et « spécifications fonctionnelles » du projet APII-SIM.

Le document des spécifications fonctionnelles présente les différents systèmes qui composent la solution proposée : l’aiguilleur, le système d’administration et le site de démonstration.

Par rapport aux objectifs fixés, l’enjeu technique essentiel se situe au niveau de la fonction de recherche. Cette fonction doit en effet préserver la qualité des solutions sans trop dégrader les performances.

La phase de conception s’organise autour du système qui implémente cette fonction de recherche, à savoir l’aiguilleur.

L’objectif des spécifications techniques est de

* définir les interfaces externes de l’aiguilleur et la méta-base
* détailler les aspects algorithmiques qui conditionnent le succès des fonctions principales (y compris la fonction de recherche d’itinéraire distribuée)

Par ailleurs les interfaces externes de l’aiguilleur et la méta-base constituent une interface d’échange d’information voyageur et intéressent à ce titre les instances de normalisation comme le GT7.2.

Les interfaces de l’aiguilleur avec les SIM (que ce soit des SIM de collectivité ou bien des SIM longues distances) feront l’objet d’un document spécifique ultérieur.

Les autres systèmes de la solution à savoir le système d’administration des méta-données et le site de démonstration ne sont pas au cœur de l’enjeu technique.

Le système d’administration des méta-données est relativement « découplé » de l’aiguilleur, la base des méta-données est finalement la seule interface entre les 2 systèmes.

Le système d’administration n’a donc pas directement d’incidence sur les objectifs de la solution globale.

Le rôle du système d’administration se limite à permettre la gestion des données durant la phase d’expérimentation du POC (3 mois). Le but du projet consiste à valider une solution technique et en évaluer les performances, le projet laisse donc au second plan les aspects spécifiques au système d’administration. Ces aspects s’inscrivent davantage dans une phase d’industrialisation. Par ailleurs, les choix de gouvernance du méta-système ont une incidence sur les fonctionnalités du système d’administration (profil des utilisateurs, droits d’accès, etc.). Les choix de gouvernance seront donc à définir avant la conception d’une version « industrialisée » du système d’administration.

Il est important de garder à l’esprit que le projet garde une part importante de recherche et développement.

La phase de conception doit éviter de figer trop tôt les choix techniques de manière à faciliter le développement en mode itératif.

Pour cette raison, le modèle de classes internes des différents systèmes ne sont pas décrits dans ce document.

Cependant ces modèles seront bien produits à la fin de la phase de développement.

# Les interfaces du système

## Les interfaces de recherche d’itinéraire et de localités

Même si l’aiguilleur (dans le cadre du POC) n’implémente pas le service de recherche de localités, celui-ci a été joint à la définition de l’interface de manière à disposer d’un ensemble de services qui soient autonomes complets et cohérents.

Dans la suite du chapitre, l’API RI simple désigne les travaux publiés sur le site de la PREDIM <http://www.predim.org/spip.php?article4057>. Cette API est conçue pour interroger un SIM pour un déplacement dans le périmètre géographique du SIM (ou au-delà par rabattement vélo ou voiture).

### Protocole

Les services mis à disposition par l’interface de l’aiguilleur correspondent à des traitements dont la durée d’exécution est difficilement prédictible. Par nature, le projet repose sur un principe de distribution de calcul auprès d’autres systèmes qui ont donc un impact variable sur les durées de traitement globales.

Comme indiqué dans le document de spécifications fonctionnelles, l’aiguilleur répond à une requête d’itinéraire en évaluant plusieurs solutions possibles, chaque solution étant répartie sur une liste de SIM spécifique.

Autant que possible, le protocole doit permettre un mode asynchrone entre l’aiguilleur et le client, et permettre ainsi à l’aiguilleur de renvoyer au client les premières solutions obtenues sans attendre l’aboutissement total de toutes les solutions.

Pour faciliter l’utilisation des interfaces de l’aiguilleur, il est préférable que le protocole permette une interrogation depuis un navigateur web directement. Autrement dit les navigateurs doivent pouvoir être des clients des interfaces.

L’API RI simple a utilisé un protocole http GET avec une sortie XML ou JSON.

Ce protocole avait été choisi pour sa simplicité d’intégration dans les navigateurs.

#### Choix de WebSocket/JSON

Au niveau du POC, il a été convenu d’utilisé WebSocket/JSON comme protocole d’implémentation des services du méta-système.

Le protocole WebSocket (dont les spécifications sont publiées sur le site <http://tools.ietf.org/html/rfc6455>) est un protocole bidirectionnel qui simplifie également l’implémentation des échanges asynchrones entre un serveur WEB et un client.

L’article suivant [http://www.WebSocket.org/quantum.html](http://www.websocket.org/quantum.html) expose assez bien les différentes solutions existantes en http par rapport aux traitements asynchrones.

Sur le plan de la sécurité des connexions, le protocole WebSocket autorise également des échanges à travers une liaison SSL.

Le protocole WebSocket s’établit en 2 phases. Une première phase de connexion en http entre le client et le serveur et une phase d’échange direct au niveau TCP.

La mise en place de ce protocole est donc à prendre en compte conjointement sur le serveur (l’aiguilleur) mais aussi sur les clients.

* Au niveau des clients WEB :
  + les dernières versions des principaux navigateurs WEB : IE, Firefox, Chrome et Safari (cf <http://caniuse.com/websockets>) intègrent nativement le protocole WebSocket sur leur moteur Javascript.
* Au niveau des serveurs WEB :
  + une série de frameworks est disponible pour faciliter l’implémentation d’un serveur WebSocket dans une variété de langages : .Net, Python, Ruby, Java, Javascript…

Une fois la connexion établie, WebSocket (https://developer.mozilla.org/en-US/docs/Web/API/WebSocket) propose 3 formats pour transmettre les données de requête au méta-serveur.

L’objet WebSocket dispose de 3 méthodes « send » chacune ayant un seul argument d’un type particulier :

* DOMString
* ArrayBuffer
* Blob

Dans la suite du document, les spécifications ne détaillent que l’usage du type DOMString et dans le cas particulier d’une chaine représentant un [objet JSON](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/JSON).

On désignera par « objet de requête » du service, l’argument fourni à la méthode send de l’objet WebSocket.

De même, les notifications des résultats de la requête seront produites sous forme d’objets JSON.

Autrement dit, la fonction qui définit l’écouteur d’événement onmessage de l’objet WebSocket, reçoit du serveur un objet event dont la propriété data est une réponse du service dans un format JSON.

Le service peut répondre à une requête en plusieurs fois.

On appellera « notification » du service, l’objet event.data reçu au niveau de l’écouteur d’événement onmessage.

#### Généralisation des implémentations du protocole

La solution technique décrite dans la suite du document et dans les XSD pourrait servir de spécification d’implémentation sur d’autres technologies que WebSocket/JSON.

Pour réutiliser les spécifications il suffit, que le protocole puisse être :

* Asynchrone
* Utiliser un format de données (entrées, sorties, exceptions) qui puisse être décrit par des structures XSD

### Les services de l’interface du méta-système

Le méta-système publie essentiellement 2 services accessibles par URL Web:

* un service de recherche de localités, associé à une url de la forme
  + « ws://[mon domaine]/1.0/search\_points »
* un service de recherche d’itinéraire, associé à une url de la forme
  + « ws://[mon domaine]/1.0/plan\_trip »

Le sous-répertoire « 1.0 » des url définit la version du protocole demandé.

#### Le service de recherche de localités

Le service est associé à une url et un protocole compatible avec WebSocket.

Ce service reçoit en entrée un objet de requête conforme à l’élément SearchPointsRequest décrit par le fichier SearchPoints.xsd.

En retour le service émet une série de notifications, chaque notification étant un objet JSON conforme à l’élément SearchPointsNotificationResponse décrit par le fichier SearchPoints.xsd.

La structure SearchPointsNotificationResponse précise pour chaque localité un identifiant (attribut id de l’élément SearchPoint) qui peut ensuite servir à former une requête au service recherche d’itinéraire (au départ ou en arrivée à cette localité).

Pour faciliter les échanges entre client et serveur :

* L’élément SearchPointsNotificationResponse rappelle l’identifiant (sur sa propriété RequestId) de la structure de requête SearchPointsRequest (attribut id)
* L’élément SearchPointsNotificationResponse indique s’il s’agit de la dernière notification (sur sa propriété isLastNotification).
* Le client peut demander au serveur de stopper la recherche de localités en cours en adressant une structure SearchPointsCancellationRequest (en précisant l’identifiant de la recherche de localités visée)
* En retour le serveur s’acquitte de cette de demande en émettant une structure SearchPointsCancellationNotificationResponse (qui précise par son identifiant quelle recherche de localités a été annulée).

#### Le service de recherche d’itinéraire

Le service est associé à une url et un protocole compatible avec WebSocket.

Ce service reçoit en entrée un objet de requête conforme à la définition de l’élément PlanTripRequest décrit par le fichier PlanTrip.xsd.

La structure PlanTripRequest permet de renseigner un départ ou une arrivée de 2 manières :

* soit en précisant une position géographique (propriété Position)
* soit en précisant l’identifiant d’une localité (propriété PlaceTypeId). Bien entendu, c’est le service de recherche de localités qui permet de se procurer un tel identifiant

En retour le service émet une série de notifications qui dépend du nombre de trace e SIM retenu pour le recherche demandée.

Le serveur envoie en principe 2 notifications pour chaque trace de SIM ayant été retenu l’algorithme de recherche (cf 5.4.2.3 Fonction de Sélection des traces pour une recherche d’itinéraire).

Pour une trace donnée, la première notification intervient au moment où le serveur a pu établir l’existence d’une solution d’itinéraire mais sans en avoir plus de détail que son horaire de départ et d’arrivée.

La première notification est transmise par une structure PlanTripExistenceNotificationResponse.

Dans le cas d’une trace de 3 SIM, cette première notification peut être adressée en fin d’étape 6 (cf cas d’une trace (A, B, C) au § 5.4.2.4.1 Détail de l’algorithme de la fonction pour un appel en « départ à »).

La seconde notification intervient à la fin de la recherche d’itinéraire pour la trace considérée, au moment où le serveur dispose de tout le détail de l’itinéraire à parcourir. Dans le cas d’une trace de 3 SIM, cette seconde notification peut être adressée en fin d’étape 13 (cf cas d’une trace (A, B, C) au § 5.4.2.4.1 Détail de l’algorithme de la fonction pour un appel en « départ à »).

La seconde notification est transmise par une structure PlanTripNotificationResponse.

Pour faciliter les échanges entre client et serveur :

* Les éléments PlanTripNotificationResponse et PlanTripExistenceNotificationResponse rappellent l’identifiant (sur leur propriété RequestId) de la structure de requête PlanTripRequest (attribut id)
* Les éléments PlanTripNotificationResponse et PlanTripExistenceNotificationResponse indiquent s’il s’agit de la dernière notification (sur sa propriété isLastNotification).
* Le client peut demander au serveur de stopper la recherche d’itinéraire en cours en adressant une structure PlanTripCancellationRequest (en précisant l’identifiant de la recherche d’itinéraire visée)
* En retour le serveur s’acquitte de cette de demande en émettant une structure PlanTripCancellationNotificationResponse (qui précise par son identifiant quelle recherche d’itinéraire a été annulée).

### La définition des structures échangées par les services

Les structures échangées par les services ont été construites à partir l’API RI Simple et en particulier à partir des définitions de la XSD de cette API.

Sur cette base de départ, un certain nombre de changements ont été introduits

* des changements pour améliorer l’interface de manière générale et sans changer sa nature
* des changements pour répondre aux besoins spécifiques du projet

#### Les améliorations apportées

Ces changements ont pour but :

* de faciliter le contrôle et la validation des structures échangées sur l’interface
* de clarifier l’usage de l’interface. Comme il s’agit d’une interface ouverte, il est souhaitable que les ré-utilisateurs puissent facilement se l’approprier

Quelques-uns de ces changements sont directement issus de recommandations venues du GT7.2, d’autres se sont inspirés de TRIAS.

De manière plus synthétique, les changements du service de recherche d’itinéraire ont porté sur

* La définition d’un espace de nommage spécifique « http://www.apiisim.fr/distributed-journey-planner/1.0 »
  + Ceci permet d’identifier les interfaces de ce projet et d’éviter toute confusion avec celles de l’API RI simple
* Une répartition de la XSD en plusieurs fichiers (cf Documents applicables), de manière à mieux séparer les définitions propres aux 2 services et à partager les structures qu’ils ont en commun :
  + le fichier PlanTrip.xsd définit le service de recherche d’itinéraire qui est associé au sous-espace « plantrip »
  + le fichier SearchPoints.xsd définit le service de recherche de localités qui est associé au sous-espace « searchpoint »
  + le fichier Protocol.xsd définit le protocole qui structure les 2 services, les définitions de ce fichier sont associées à l’espace de nommage « http://www.apiisim.fr/distributed-journey-planner/1.0/protocol »
  + le fichier Site.xsd définit la représentation des points de passage, les définitions de ce fichier sont associées à l’espace de nommage « http://www.apiisim.fr/common/1.0/site »
  + le fichier Itinerary.xsd définit la représentation d’itinéraires, les définitions de ce fichier sont associées à l’espace de nommage « http://www.apiisim.fr/common/1.0/itinerary »
  + le fichier Itinerary-Request.xsd définit des éléments de requête d’itinéraire, les définitions de ce fichier sont associées à l’espace de nommage « http://www.apiisim.fr/common/1.0/itinerary-request »

Les changements au niveau du service de recherche d’itinéraires ont porté sur

* Une distinction des modes pour différencier ceux qui appartiennent au domaine du transport en commun (PublicTransportModeEnumeration) et des autres modes (SelfDriveModeEnumeration). Le critère retenu pour départager les modes de transport entre ces 2 énumérations est le suivant : si le mode de transport permet au voyageur d’effectuer son itinéraire sans avoir à se diriger par lui-même, il s’agit d’un mode de transport public.
  + PTRideType est donc associé aux modes SelfDriveModeEnumeration
  + LegType est donc associé aux modes PublicTransportModeEnumeration
* La structure SiteTypeEnumeration a été étendue de manière à représenter une position géographique quelconque (LOCATION) ou bien un tronçon de voirie (ROAD\_LINK).
  + De la même manière que l’interface a prévu de sélectionner en départ ou arrivée une commune complète (pour permettre une recherche du type Paris / Marseille), on peut imaginer un usage équivalent pour aller d’une rue à une autre.
* La structure SiteTypeType est complétée d’un élément POITypeName. Cet élément est utile pour adapter la présentation dans le cas où SiteTypeType représente un POI. Cette information supplémentaire peut servir de critère de style (pour associer une icône par exemple).
* Pour faciliter la lecture des itinéraires, les structures PTRideType, LegType et TripType disposent désormais de nouveaux éléments qui sont des raccourcis comme le premier / dernier point du parcours, la distance parcourue…
* Pour faciliter la lecture des horaires sur les parcours qui s’étalent sur plus d’une journée calendaire, tous les horaires sont désormais dans un type « dateTime ».
* Sur la description de parcours en transport en commun (PTRideType), il est précisé s’il s’agit du dernier service de transport assuré sur la ligne et desservant l’arrêt au départ du tronçon de ligne à emprunter.
* Sur la description de parcours en transport en commun (PTRideType), il est indiqué si le tronçon à parcourir subit une perturbation ou non.

Les changements du service de recherche de localités ont porté sur

* la prise en compte de la langue dans la structure de requête. La toponymie peut effectivement varier : « Londres » et « London », « Naples » et « Napoli ».
* la prise en compte d’une emprise géographique au niveau de l’élément GeoRestriction. L’information géographique est souvent disponible au niveau du client (c’est bien intégré aujourd’hui dans les navigateurs).
* la prise en compte d’un score. Le score est en quelque sorte une fonction qui associe chaque localité retournée à une valeur numérique. Les valeurs numériques représentent le degré de pertinence des localités associées par rapport à la saisie de l’utilisateur. Pour calculer une valeur numérique relative à la saisie utilisateur, le service peut prendre en compte toutes les caractéristiques disponibles de la localité : son nom, la phonétisation du nom, la commune (code postal et nom de commune), etc… Dans la mesure où le service de recherche retourne des localités de nature différente (POI, arrêt, adresse), il est indispensable d’avoir un critère commun d’évaluation de pertinence de la proposition de localité par rapport aux critères fournis.

#### Les changements liés aux exigences fonctionnelles du projet

##### Prise en compte du protocole

Par rapport à l’API RI simple qui formule les critères de recherche sous forme de paramètres d’url, le protocole WebSocket conduit à définir la requête sous d’objet JSON

Le service de recherche d’itinéraires reçoit une requête sous forme d’objets JSON (PlanTripRequest) et produit une réponse sous forme de notifications (objet JSON PlanTripNotificationResponse).

Le service de recherche de localités reçoit une requête sous forme d’objets JSON (SearchPointsRequest) et produit une réponse sous forme de notifications (objet JSON SearchPointsNotificationResponse).

De plus, un identifiant (sous forme d’attribut « id ») est ajouté à chaque structure de requête. Cet identifiant est destiné à être rappelé par les structures de réponse au niveau de l’élément RequestId.

En procédant ainsi, un client est en mesure de gérer l’envoi et la réception de plusieurs recherches d’itinéraire en même temps. Lorsque le client reçoit une notification de réponse, l’identifiant de requête permet au client de savoir à quelle requête correspond la structure de réponse reçue.

Pour faciliter la gestion de l’attente des notifications de réponse au niveau du client, les notifications précisent si d’autres notifications sont à venir ou non. Ces précisions figurent aussi bien sur l’élément PlanTripNotificationStatus que l’élément SearchPointNotificationStatus.

##### La liste des modes de transport

L’API RI simple référence les modes de transport du projet de normalisation européen TRIDENT. Il semble plus indiqué aujourd’hui de s’appuyer sur les spécifications du projet de norme européenne NeTEx :

* NeTEx couvre un périmètre fonctionnel plus vaste que TRIDENT
* NeTEx est à un stade plus avancé du processus de normalisation fixé par le CEN, il s’agit désormais d’une *Technical Specification*.

La liste des modes définie sur l’interface du service de recherche se base sur les modes définis par l’énumération proposée par NeTEx, toutefois

* le mode « selfDrive » de NeTEx est remplacé par les modes « bike » et « car ».
* le mode « unknown » n’a pas d’intérêt au niveau des critères de recherche, ce mode est seulement en restitution d’une réponse. Les énumérations de mode au niveau des structures LegType et PTRideType disposent bien du mode « unknown ».

###### Distinction entre les modes définis sur l’interface et le service à l’utilisateur final

Sur l’interface d’appel du service de recherche (structure PlanTripRequest), la finalité de la liste des modes de transport (TransportModeEnumeration) est de traduire les choix qui sont proposés à l’utilisateur final par service de recherche d’itinéraire.

Autrement dit, la liste doit être assez riche pour séparer les modes qui sont perçus comme différents dans l’esprit de l’utilisateur final.

Pour autant la liste des modes de transport sur l’interface n’a pas besoin d’être calquée sur ce que présente un formulaire de recherche d’itinéraire.

Entre l’interface du service de recherche et le formulaire c’est l’application le de formulaire qui pilote aussi la correspondance entre :

* les modes de transport proposés à l’utilisateur final
* les modes de transport définis dans la requête au service de recherche

On peut imaginer un formulaire de recherche qui propose le mode « bus » sans proposer le mode « trolley bus ». Ainsi lorsque l’utilisateur sélectionne le mode « bus », la requête adressée à l’interface de l’aiguilleur peut faire apparaître les modes « bus » et « trolley bus » dans la structure PlanTripRequest.

Il va de même des modes « rail », « urbanRail », « intercityRail ».

##### Choix de plusieurs modes

Au niveau de la structure de requête de recherche d’itinéraires, la cardinalité des éléments Mode a été revue pour permettre la saisie d’une liste de modes.

##### Identification des SIM sur les itinéraires partiels

La recherche d’itinéraire est conçue pour proposer des itinéraires qui proviennent de SIM différents.

La séquence partialTrips liste un descriptif des itinéraires partiels. Chaque descriptif est identifié et précise quel est le système ayant fourni l’itinéraire partiel.

L’itinéraire global (structure TripType) précise au niveau de ses sections à quel itinéraire partiel se rapporte chaque section.

Les sections qui décrivent le déplacement au niveau d’une transition ne se rapportent à aucun itinéraire partiel.

Toutes les autres sections doivent obligatoirement renseigner leur itinéraire partiel d’appartenance.

##### Le TAD

Selon les offres de transports locales et les possibilités des SIM, les solutions proposées peuvent faire appels à des lignes de Transport Public régulières comme à des lignes liées à une offre de Transport à la Demande.

Le TAD peut être perçu comme une particularité de la ligne indépendante du mode.

La structure ODTInformation (optionnelle) permet de décrire les particularités du TAD sur un PTRide.

##### Le co-voiturage

Selon les offres de transports locales et les possibilités des SIM, les solutions peuvent proposer des offres de co-voiturage.

Dans ce cas, l’élément PTRide renseigne une structure CarPoolInformationType.

##### Les rabattements

Comme indiqués dans le document de spécifications fonctionnelles, le rabattement peut être proposé dès que l’utilisateur demande au moins un mode en transport en commun et un autre type de mode (voiture ou vélo).

Au niveau de la requête au service de recherche, il est possible de préciser pour chacun des modes hors transport en commun (voiture ou vélo) s’il se limite au début ou la fin du parcours.

C’est le rôle de l’élément TripPart.

## Le modèle des méta-données

Les tables et les vues du modèle de la base des méta-données sont présentées dans ce chapitre.

#### MIS

Table des SIM (Système d’Information Multimodale : Multimodal Information System en anglais) auxquels le système peut se connecter.

La table contient donc

* les informations nécessaires pour se connecter aux interfaces génériques d’accès aux SIM (cf 5.3 Les interfaces génériques requises sur les SIM)
* les dates qui définissent la période pour laquelle le SIM est en mesure de réaliser des calculs d’itinéraire
* une emprise géographique du SIM
* des champs qui décrivent les différentes capacités du calculateur de chaque SIM

Les capacités du SIM peuvent éventuellement figurer sur l’interface générique comme fonction de consultation. Lors de la phase de développement, les champs liés à la description du SIM pourront éventuellement être adaptés aux contraintes rencontrées.

|  |
| --- |
| MIS |
| *(Clé)* MisId | Id interne | Int compteur |
| Name | Nom du SIM | Varchar(50) |
| Comment | Commentaires éventuels sur le SIM | Varchar(255) |
| APIUrl | Adresse du serveur du SIM diffusant l’API de requêtes partielles. | Varchar(255) |
| APIKey | Clé d’authentification de l’API. | Varchar(50) |
| StartDate | Date de début de l’offre gérée sur le SIM | Date |
| EndDate | Date de fin de l’offre gérée sur le SIM | Date |
| MultipleStartsAndArrivals | Possibilité de faire une recherche d’itinéraire n-m non détaillée | Int |
| GeographicPositionCompliant | Possibilité de faire une recherche d’itinéraire avec une position géographique comme point de départ ou arrivée | Boolean |

La colonne MultipleStartsAndArrivals est importante au niveau de la sélection des traces. Les SIM qui ne gèrent pas les recherches n-m doivent nécessairement se trouver en début ou en fin de trace (cf 6.4.2.3 Fonction de Sélection des traces pour une recherche d’itinéraire).

#### MODES

Table des modes disponibles dans le système. La table contient les mêmes entrées que l’énumération TransportModeEnumeration définie sur l’interface de l’aiguilleur. Le contenu de cette table est susceptible d’évoluer pour de l’implémentation.

|  |
| --- |
| MODES |
| *(Clé)* ModeId | Id interne | Int (compteur) |
| ModeCode | Chaîne de caractères indiquant le mode | Varchar(50) |

#### MIS\_MODE

Table de jointure décrivant quels modes sont disponibles sur chaque SIM. Cette table est utile pour ne pas appeler un SIM ne pouvant pas répondre aux contraintes de la requête (par exemple on ne veut pas de ferré mais le SIM ne contient que du ferré).

|  |
| --- |
| MIS\_MODE |
| *(Clé)* MisId | Id du SIM | Int |
| *(Clé)* ModeId | Id du mode | Int |



#### STOPS

Ensemble des arrêts de chaque SIM avec leur géolocalisation.

|  |
| --- |
| STOPS |
| *(Clé)* StopId | Id de l’arrêt | Int (compteur) |
| *(Idx)* StopCode | Id de l’arrêt dans le référentiel du SIM | Varchar(50) |
| *(Clé)* MisId | Id du SIM auquel appartient l’arrêt | Int |
| Name | Nom de l’arrêt (utile pour le debug) | Varchar(255) |
| Type | Type d’arrêt. Le type fait référence ici à une codification numérique de manière à ne pas être trop lié à une classification.  selon la définition donnée par le « modèle d’arrêt partagé » (énum : GL (Groupe de lieu), LAMU (Lieu d’arrêt multimodal), ZE (Zone d’embarquement) etc…) | Int |
| AdministrativeCode | Identifiant administratif de la commune d’appartenance | Varchar(255) |
| StopParentId | Référence vers l’arrêt de niveau hiérarchique supérieur | Int |
| TransportMode | Mode de transport principal  Les valeurs correspondent à l’énumération TransportModeEnumeration définie dans Itinerary.xsd | Varchar(255) |
| QuayType | Type de zone d’embarquement.  Les valeurs correspondent à l’énumération Netex QuayTypeEnumeration. | Varchar(255) |
| Lat | Latitude WGS84 | Float |
| Long | Longitude WGS84 | Float |

La propriété Type de l’arrêt fait référence à une codification numérique de manière à ne pas être trop lié à une classification. En procédant ainsi, il est possible de représenter par exemple, les types d’arrêt du modèle d’arrêt partagé.

Dans le cas d’une alimentation basée sur un référentiel qui implémente le modèle d’arrêt partagé, un certain nombre de propriétés peuvent être définies

* Type : le type d’arrêt d’arrêt par rapport à une hiérarchique de l’arrêt (lieu d’arrêt Multimodal, lieu d’arrêt Monomodal, etc)
* StopParentId : la référence vers l’arrêt de niveau hiérarchique supérieur
* AdministrativeCode : le code INSEE de la commune d’appartenance
* TransportMode : le mode de transport principal de l’arrêt
* QuayType : le type de quai

La hiérarchie d’arrêt (définie à travers StopParentId et Type) est à prendre en compte au niveau du backoffice pour gérer la création et la mise à jour des transitions (cf 5.5.1.3 Fonction de mise à jour des transitions). Les arrêts appartenant à une même hiérarchie devraient être mis en relation dans la table CONNECTION (sauf exception particulière).

Dans le cadre du POC, seuls les arrêts (de type arrêts physiques) sont collectés.

Aucune des 5 propriétés ci-dessus n’est utilisée.

Cette table doit être Indexée par StopCode et MisId pour accélérer la construction de la table CONNECTIONS.

Au niveau physique les colonnes Lat et Long peuvent être représentés par un champ géographique sur lequel un Index spatial peut accélérer la sélection des SIM à partir d’un rayonnement autour d’une position géographique (cf 5.4.2.2 Fonction de sélection des SIM à proximité d’une position géographique).

#### CONNECTIONS

Table des correspondances possibles entre les arrêts de SIM différents. Le contenu de la table obtenue par rapprochement géographique des éléments de la table STOPS.

Dans le cas d’une alimentation basée sur un modèle d’arrêt partagé, le regroupement d’arrêts dans une hiérarchie (arrêts d’un lieu d’arrêt Mulitmodal par exemple) doit se traduire par des connections entre tous les arrêts de la hiérarchie.

|  |
| --- |
| CONNECTIONS |
| *(Clé)* StopId1 | Id de l’arrêt dans la table STOP | Int |
| *(Clé)* StopId2 | Id de l’arrêt dans la table STOP | Int |
| Distance | Distance à vol d’oiseau (m) | Int |
| Duration | Temps estimé de correspondance (min) | Int |
| PrmDuration | Temps estimé de correspondance pour une PMR (min) | Int |

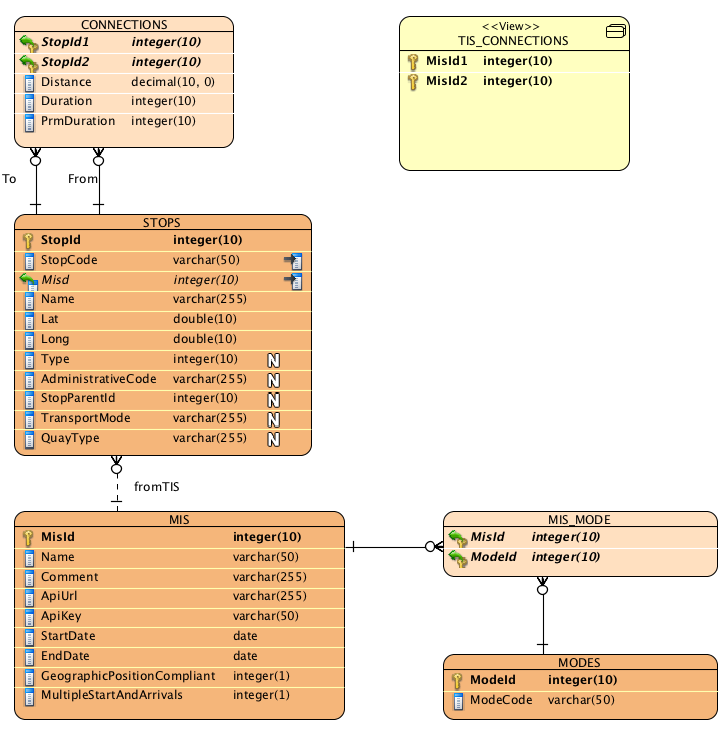
#### MIS\_CONNECTIONS

Vue directement déduite de la table CONNECTIONS indiquant quelles SIM sont connectés entre eux et à quel point (proportion d’arrêts « en commun »).

Selon l’implémentation de l’aiguilleur, il n’est pas forcément indispensable de stocker les données de MIS\_CONNECTION sous forme de table.

|  |
| --- |
| MIS\_CONNECTIONS |
| *(Clé)* MisId1 | Id du SIM 1 connecté au SIM 2 | Int |
| *(Clé)* MisId2 | Id du SIM 2 connecté au SIM 1 | Int |

#### Schéma de la base des méta-données



Le symbole N signale les champs pouvant ne pas être définis.

## Les interfaces génériques requises sur les SIM

Ces interfaces sont très importantes puisqu’elles constituent les prérequis à satisfaire par les SIM pour être intégrés au projet global de calcul distribué.

Bien que les interfaces génériques fassent l’objet d’un document ultérieur, les fonctionnalités requises sont résumées succinctement ci-dessous :

**Appel RI détaillée *1-n***

On appelle « Appel RI détaillé » un appel *1-n* à une SIM qui a comme paramètres de requête :

* Un ou n arrêts de départ dans le référentiel du SIM, et pour chaque arrêt une durée d’accès spécifique
  + Si cette liste contient plus d’un arrêt, la liste des points d’arrivée ne contient qu’un seul point
* Un ou n arrêts d’arrivée dans le référentiel du SIM, et pour chaque arrêt une durée d’accès spécifique
  + Si cette liste contient plus d’un arrêt, la liste des points de départ ne contient qu’un seul point
* Une liste de modes
* Une contrainte sur les modes « selfDrive » (vélo, voiture) pour lesquels on peut autoriser un rabattement en début ou fin de parcours
* Un horaire de départ (*départ à*) OU un horaire d’arrivée (*arrivée à*)

Et dont la réponse est :

* La description détaillée du meilleur itinéraire qui permet de définir le meilleur arrêt parmi les *n* possibles (au départ ou à l’arrivée selon le cas)
* En *départ à* : le meilleur itinéraire est celui qui arrive au plus tôt.
* En *arrivée à* : le meilleur itinéraire est celui qui part au plus tard.

Cette fonction est nécessaire à l’aiguilleur (cf 6.4 Le système Aiguilleur).

**Appel RI non détaillée *n-m***

On appelle « Appel RI non détaillé » un appel *n-m* à une SIM qui a comme paramètres de requête :

* *n* arrêts de départ dans le référentiel du SIM, et pour chaque arrêt une durée d’accès spécifique
* *m* arrêts d’arrivée dans le référentiel du SIM, et pour chaque arrêt une durée d’accès spécifique
* Une liste de modes
* Une contrainte sur les modes « selfDrive » (vélo, voiture) pour lesquels on peut autoriser un rabattement en début ou fin de parcours
* Un horaire de départ (*départ à*) OU un horaire d’arrivée (*arrivée à*)

Et dont la réponse est :

* La liste d’itinéraires possibles sans leur détail
  + La liste contient au plus n itinéraires en « départ à », au plus m itinéraires en « arrivée à »
  + La liste peut compter moins d’itinéraires s’il n’existe pas de solution pour certains couples d’arrêt de départ et d’arrivée.
* En *départ à* :
  + les horaires d’arrivée des itinéraires sont les meilleurs possibles (arrivée au plus tôt possible à leur point de destination)
  + les horaires de départ sont postérieurs ou égal à celui de la requête et tiennent compte des durées d’accès aux arrêts de départ
* En *arrivée à :*
  + les horaires de départ des itinéraires sont les meilleurs possibles (départ au plus tard possible de leur point de départ)
  + les horaires d’arrivée sont antérieurs ou égal à celui de la requête et tiennent compte des durées d’accès aux arrêts d’arrivée

Cette fonction est nécessaire à l’aiguilleur (cf 6.4 Le système Aiguilleur).

**Appel RI non détaillée optimisée *1-n***

On appelle « Appel RI non détaillée optimisée » un appel *1-n* à une SIM qui a comme paramètres de requête:

* Un ou *n* arrêts de départ dans le référentiel du SIM, et pour chaque arrêt une durée d’accès spécifique
  + Si cette liste contient plus d’un arrêt, la liste des points d’arrivée ne contient qu’un seul point
* Un ou *n* arrêts d’arrivée dans le référentiel du SIM, et pour chaque arrêt une durée d’accès spécifique
  + Si cette liste contient plus d’un arrêt, la liste des points de départ ne contient qu’un seul point
* Une liste de modes
* Eventuellement, une contrainte sur les modes « selfDrive » (vélo, voiture) pour lesquels on peut autoriser un rabattement en début ou fin de parcours
* Un horaire de départ (*départ à*) OU un horaire d’arrivée (*arrivée à*)

Et dont la réponse est :

* La liste des *n* meilleurs itinéraires possible sans leurs détails
  + La liste peut compter moins d’itinéraires s’il n’existe pas de solutions à destination ou en provenance de certains arrêts
* En *départ à*, les horaires de départ des itinéraires sont postérieurs ou égal à l’horaire demandé et tiennent compte des durées d’accès aux points de départ
* En *arrivée à,* les horaires d’arrivée des itinéraires sont antérieurs ou égal à l’horaire demandé et tiennent compte des durées d’accès aux points d’arrivée
* Les itinéraires proposent les meilleurs horaires d’arrivée possible (l’horaire d’arrivée le plus tôt possible) au point de destination (ou aux points de destinations)
* Les itinéraires proposent les meilleurs horaires possibles au départ (l’horaire de départ le plus tardif possible) du point de départ (ou des points de départ)

Au niveau du SIM, si le service de calcul d’itinéraire interne ne permet pas d’obtenir en un seul calcul ces solutions, le composant « traducteur » déployé au niveau du SIM peut procéder en 2 appels.

Par exemple, pour une requête en Départ à,

* le 1er appel détermine le meilleur horaire d’arrivée possible (avec calcul en Départ à)
* le 2ème appel détermine le meilleur horaire de départ possible (avec calcul en Arrivée à l’horaire obtenu en résultat du 1er appel)

Cette fonction est nécessaire à l’aiguilleur (cf 6.4 Le système Aiguilleur).

Par rapport à la fonction « Appel RI non détaillée n-m », cette fonction introduit une contrainte sur les réponses au niveau de l’horaire d’arrivée et de l’horaire de départ.

**Sélection de l’ensemble des arrêts**

On appelle « Sélection de l’ensemble des arrêts » un appel à une SIM dont la réponse est :

* La totalité des arrêts du SIM

La description des arrêts du SIM précise leur identifiant dans le SIM, leur position géographique, leur identifiant national (s’il existe), le nom de l’arrêt et le type d’arrêt.

Cette fonction est nécessaire au système d’administration (cf 6.5 Architecture du système d’administration).

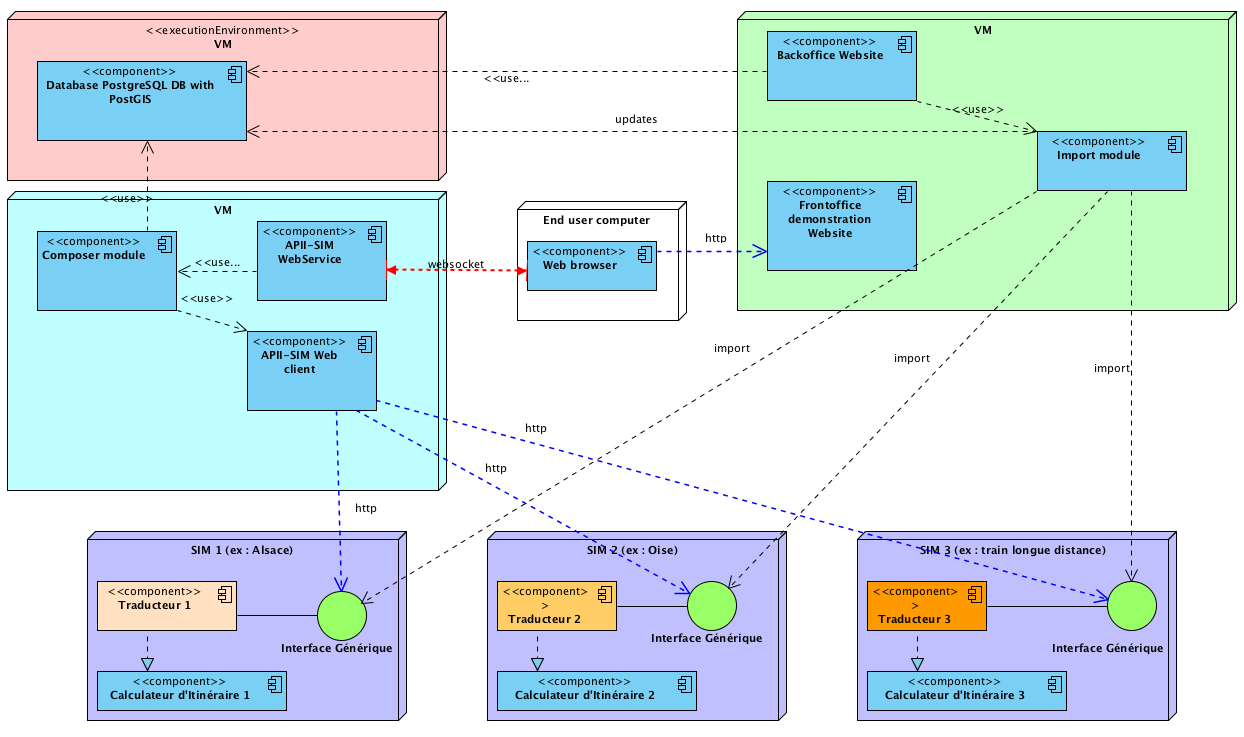
A cette liste de fonctionnalités, un certain nombre de fonctions de consultation des capacités du SIM pourront aussi être ajoutées si nécessaire.

Ces fonctions de consultation ne sont pas destinées à l’aiguilleur a priori. En effet il est préférable d’éviter le nombre d’échange entre l’aiguilleur et les SIM.

Par contre ces fonctions peuvent éventuellement être utiles au système d’administration pour la mise à jour des méta-données qui décrivent les SIM (cf 5.2.1.1 MIS).

## Les interfaces dans l’architecture de déploiement

Le schéma ci-dessous présente un diagramme de déploiement des différents composants de la solution. Le schéma fait apparaître les interfaces d’échanges entre les composants.



# Les spécifications techniques générales des systèmes de la solution

## Définition du graphe de SIM

Le document de spécifications fonctionnelles définit le sens donné au terme SIM dans le cadre de ce projet. Ce terme englobe les SIM d’agglomération (ou les SIM régionaux) traditionnels comme les calculateurs longues distances.

Lorsqu’on observe la distribution des arrêts de SIM différents dans l’espace, on peut remarquer la proximité entre certains arrêts de SIM distincts. C’est cette proximité qui permet d’établir des liaisons (appelées Transitions) entre arrêts.

Le schéma ci-dessous illustre le cas de 5 SIM traditionnels et de 2 calculateurs longues distances.

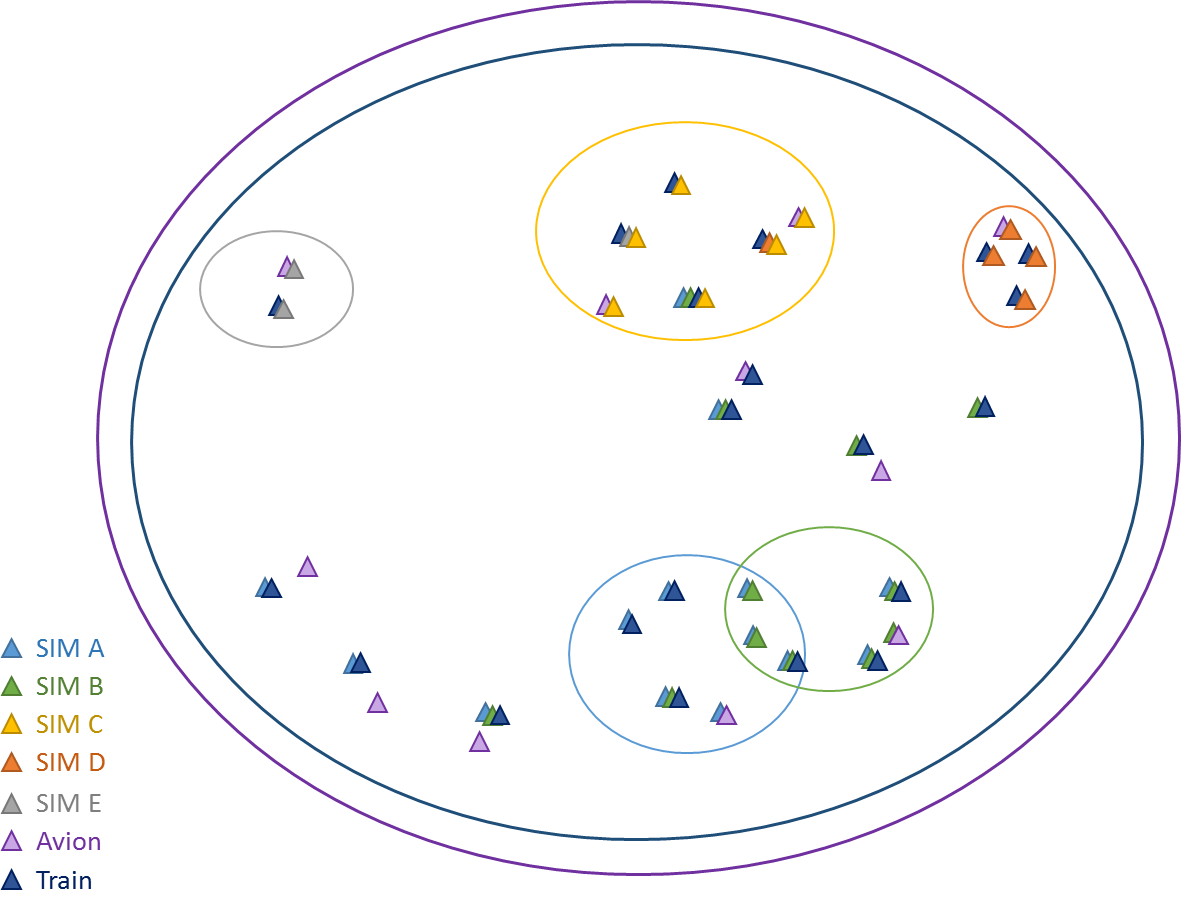


Figure 1 Répartition géographique des arrêts des SIM

2 SIM distincts sont dits connectés, lorsqu’il existe au moins un arrêt dans chaque SIM respectif qui définisse une transition. Le graphe de SIM représente l’ensemble de connexions entre SIM.

Le schéma ci-dessous présente un tel graphe.

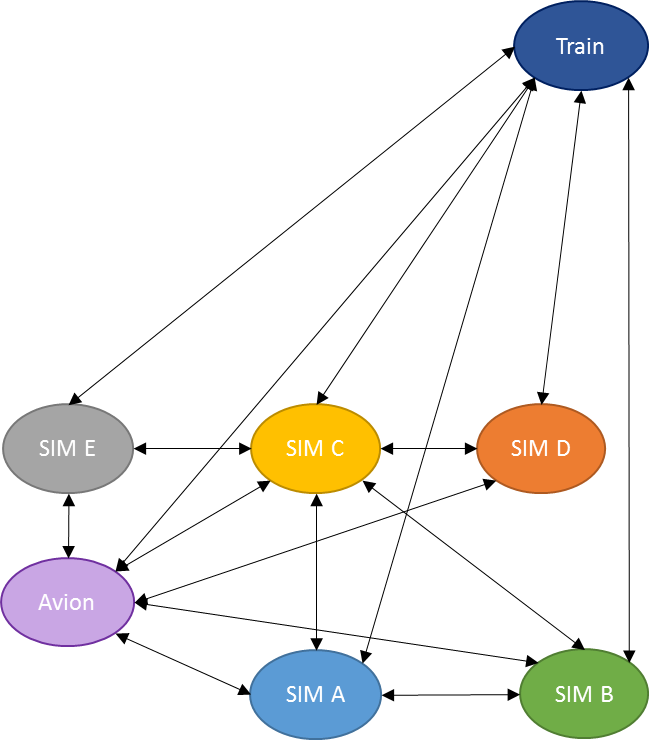


Figure 2 Graphe de SIM

## Définition de l’itinéraire partiel

Comme indiqué au chapitre consacré aux interfaces, l’aiguilleur dispose d’une interface d’échange avec les SIM. Cette interface dispose d’une fonction de recherche d’itinéraires.

Les réponses que fournissent les SIM à l’aiguilleur à travers cette interface s’appellent les itinéraires partiels. Cette définition reprend celle de Delfi.

## Notations

Pour deux SIM A et B, on notera A⋂B l’ensemble des arrêts de A qui ont une correspondance avec au moins un arrêt de B.

La relation n’est pas symétrique puisque A et B sont complètement distincts : les arrêts de B ne sont pas les même que ceux de A (la notation intersection est abusive).

Pour un arrêt *a* dans une SIM A et une SIM B, on notera B⋂*a* l’ensemble des arrêts de B qui ont une correspondance avec a.

## Le système Aiguilleur

### Introduction

Le système d’aiguilleur met à disposition les interfaces WebSockets du service de recherche d’itinéraires.

L’aiguilleur est en interface avec la base des méta-données et l’ensemble des interfaces génériques publiées par chacun des SIM.

C’est la base des méta-données qui permet à l’aiguilleur de disposer des informations de connexion aux différents SIM (cf table MIS avec les colonnes APIUrl, APIKey).

### Spécification des principales fonctions

#### Fonction d’évaluation du bilan carbone

Cette fonction reçoit en entrée

* la liste des bilans carbone de l’ensemble des itinéraires partiels

Cette fonction renvoie en sortie

* le bilan carbone global

Dans le bilan global se calcule

* le bilan carbone effectif global est obtenu par addition des bilans effectifs partiels
* le bilan carbone équivalent voiture global est obtenu par addition des bilans équivalents voiture partiels
* le ratio bilan effectif sur équivalent voiture est recalculé avec les 2 résultats ci-dessus

La fonction d’évaluation ne produit pas de résultat si un SIM ou plus ne fournit pas le bilan carbone de son itinéraire partiel.

#### Fonction de sélection des SIM à proximité d’une position géographique

Cette fonction reçoit en entrée

* la position géographique du point de départ
* la date associée à la requête de recherche d’itinéraire

Cette fonction renvoie en sortie

* une liste de SIM.

Les SIM sélectionnés sont ceux qui ont

* au moins un arrêt situé à vol d’oiseau en deçà d’un seuil paramétrable (a priori le seuil est évalué par rapport à un déplacement en marche à pied)
* une offre valide à la date demandée

#### Fonction de Sélection des traces pour une recherche d’itinéraire

Cette fonction reçoit en entrée

* la position géographique du point de départ (noté loc1)
* la position géographique du point d’arrivée (noté loc2)
* les modes souhaités

Cette fonction renvoie en sortie

* une sélection de *k* listes de SIM
  + il s’agit des *k* plus courts chemins entre les SIM
  + ces plus courts chemins s’appellent les traces de SIM
* Le point de départ est contenu dans le premier SIM de la trace.
* Le point d’arrivée est contenu dans le dernier SIM de la trace.
* Tous les SIM proposés dans les chemins sont compatibles avec au moins l’un des modes souhaités.
* La trace contient au maximum un seul SIM ne pouvant pas traiter un appel « RI non détaillée *n-m* » et ce SIM est situé en première ou dernière position de la trace. La capacité du SIM à réaliser des recherches d’itinéraire non détaillées n-m est décrite dans les méta-données (cf 5.2.1.1 MIS).

On appelle taille d’une trace le nombre de SIM dans la trace.

La taille maximum des traces de SIM doit être paramétrable. Ce paramètre est limité à 3 dans le cadre du POC.

##### Détail de l’algorithme de la fonction et de sa complexité

Lorsque l’aiguilleur reçoit une requête d’itinéraire de loc1 vers loc2, l’aiguilleur trouve les SIM où sont situés loc1 et loc2 en appelant la fonction de « Sélection des SIM à proximité d’une position géographique ».

Le traitement consiste ensuite en une recherche de plus court chemin dans le graphe des SIM dans lequel

* les nœuds seraient les SIM
* les arcs les seraient les connexions entre SIM

Le nombre de nœuds et d’arcs étant très limité, l’algorithme ne devrait pas être trop gourmand en temps de calcul, bien qu’on lui demande de calculer tous les chemins possibles distincts sans boucle de longueur inférieure au maximum fixé comme paramètre (3 dans le cadre du POC).

##### Illustration de la fonction sur un exemple

En repartant du schéma de la Figure 2, supposons que la fonction reçoive 2 positions géographiques :

* La position de départ située dans les SIM A et SIM Train
* La position d’arrivée située dans le SIM D
* Les modes Train et Avion et associés aux SIM A et SIM D

La fonction recherche les chemins suivants de (SIM A, SIM Train) vers SIM D.

La fonction renvoie la liste suivante des traces de SIM :

* SIM Train -> SIM D
* SIM A -> SIM C-> SIM D
* SIM A -> SIM Train -> SIM D (il est possible d’aller à une meilleure gare avec le réseau A pour prendre un train plus tôt vers le SIM D)
* SIM A -> SIM Avion -> SIM D

Conformément au paramètre qui fixe la taille maximum des traces de SIM, les trajets à 4 SIM et plus ne sont pas considérés (ceux-ci sont beaucoup plus longs à traiter).

#### Fonction de distribution des calculs de RI pour une trace donnée

Cette fonction reçoit en entrée

* la position géographique du point de départ (notée loc1)
* la position géographique du point d’arrivée (notée loc2)
* une trace de SIM
* une liste de modes souhaités
* un horaire de départ ou d’arrivée (l’horaire fournit aussi la date)

Cette fonction renvoie en sortie

* un itinéraire détaillé allant de loc1 à loc2

L’itinéraire renvoyé

* respecte la liste des modes souhaités
* prend en compte l’horaire de départ ou d’arrivée
* agrège des solutions partielles d’itinéraires fournies par les différents SIM de la trace de SIM

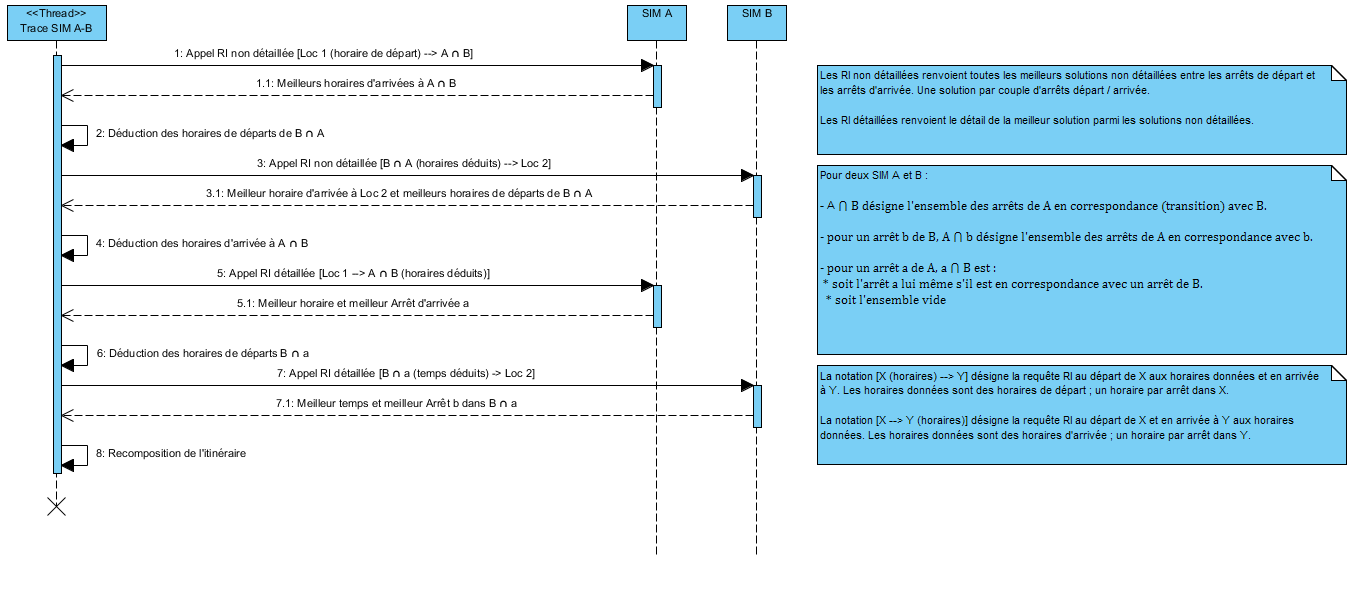
##### Détail de l’algorithme de la fonction pour un appel en « départ à »

L’aiguilleur utilise l’interface de RI des différents SIM avec une série d’appels que ce soit des « Appels RI non détaillées » ou des « Appels RI détaillées ».

Dans le cas d’une trace (A, B) et d’un horaire en « départ à », l’aiguilleur effectue les actions suivantes, dans cet ordre :

1. ***Appel RI non détaillée n-m*** à A en *départ à* de (loc1) vers A⋂B avec l’horaire de départ fourni par la requête (n=1 ici). Obtention des meilleurs horaires d’arrivée pour chaque arrêt de A⋂B.
2. Déduction des meilleurs horaires d’arrivée à B⋂A en rajoutant les durées de correspondance sur les transitions.
3. ***Appel RI non détaillée optimisée 1-n***à B en *départ à* de B⋂A vers loc2, avec les horaires de départ précédents. Obtention du meilleur horaire d’arrivée possible à loc2 et des meilleurs horaires de départ de B⋂A.
4. Déduction des horaires de départ au plus tard de A⋂B vers loc2, cette déduction utilise les horaires de départ des réponses de l’étape 3 et retire les durées de correspondance sur les transitions.
5. ***Appel RI détaillée 1-n***à A en *arrivée à* de (loc1) vers A⋂B en utilisant les horaires précédents. On obtient donc le meilleur arrêt a de A⋂B.
6. Déduction des horaires de départ aux arrêts B⋂a en ajoutant les durées de correspondance sur les transitions entre a et B⋂a.
7. ***Appel RI détaillée 1-n***à B *en départ à* de B⋂a vers loc2. On obtient donc le meilleur arrêt b de B⋂a
8. Recomposition de l’itinéraire de loc1 vers loc2 à partir des deux résultats détaillés précédents.

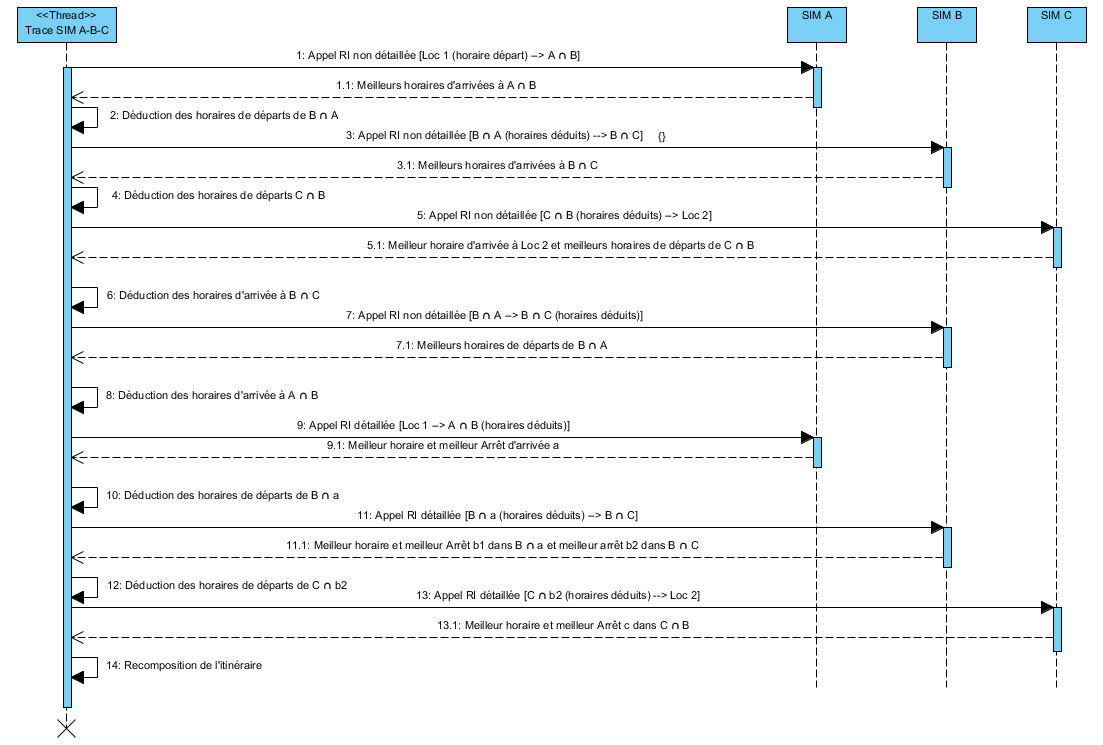
Le schéma ci-dessous présente le diagramme de séquence qui correspond à cet algorithme



Dans le cas d’une trace (A, B, C) et d’un horaire de départ, l’aiguilleur effectue les actions suivantes, dans cet ordre :

1. ***Appel RI non détaillée n-m*** à A en *départ à* de (loc1) vers A⋂B avec l’horaire de départ fourni par la requête (n=1 ici). Obtention des meilleurs horaires d’arrivée pour chaque arrêt de A⋂B.
2. Déduction des meilleurs horaires d’arrivée à B⋂A en rajoutant les durées de correspondance sur les transitions.
3. ***Appel RI non détaillée n-m*** à B en *départ à* de B⋂A vers B⋂C, avec comme horaires de départ ceux obtenus à l’étape 2.
4. Déduction des meilleurs horaires d’arrivée à C⋂B en rajoutant les durées de correspondance sur les transitions.
5. ***Appel RI non détaillée optimisée 1-n*** à C en *départ à* de C⋂B vers loc2, avec comme horaires de départ ceux obtenus à l’étape 4. Obtention du meilleur horaire d’arrivée possible à loc2 et des meilleurs horaires de départ de C⋂B.
6. Déduction des horaires de départ au plus tard de B⋂C vers loc2. L’évaluation se fait comme suit : les horaires obtenus à l’étape 5 sont diminués des durées de correspondance sur les transitions.
7. ***Appel RI non détaillée n-m***à Ben *arrivée à* de B⋂A vers B⋂C en utilisant les horaires précédents.
8. Déduction des horaires de départ au plus tard de A⋂B vers B⋂C. L’évaluation se fait comme suit : les horaires obtenus à l’étape 7 sont diminués des durées de correspondance sur les transitions.
9. ***Appel RI détaillée 1-n***à A en *arrivée à* de loc1 vers A⋂B avec comme horaires ceux de l’étape précédente. On obtient donc le meilleur arrêt a de A⋂B.
10. Déduction de l’horaire de départ aux arrêts B⋂a en ajoutant leurs durées de correspondance sur les transitions entre a et B⋂a.
11. ***Appel RI détaillée 1-n***à Ben *départ à* de B⋂a vers B⋂C. On obtient donc le meilleur arrêt de B⋂a (b1) et le meilleur arrêt de B⋂C (b2).
12. Déduction de l’horaire de départ aux arrêts C⋂b2 en ajoutant les durées de correspondance sur les transitions entre b2 et C⋂b2.
13. ***Appel RI détaillée 1-n***à Cen *départ à* de C⋂b2 vers loc2. On obtient donc le meilleur arrêt c de C⋂b2.
14. Recomposition de l’itinéraire de loc1 vers loc2 à partir des trois résultats détaillés précédents.

Le schéma ci-dessous présente le diagramme de séquence qui correspond à cet algorithme



Les schémas ci-dessous donnent une représentation spatiale des données échanges au cours de l’algorithme dans le cas d’une trace (A, B, C).

Le schéma ci-dessous illustre les appels RI pour les étapes 1, 3 et 5.

a1 à a6 sont les points de A⋂B, b1 à b3 sont des points de B⋂A, b4 à b6 sont des points de B⋂C, c1 à c3 sont les points de C⋂B

Dans ce schéma, les arrêts de A⋂B n’établissent chacun qu’une seule transition sur B⋂C, même chose pour les transitions entre B et C.

t0 correspond à l’horaire de la requête en paramètre d’entrée de l’algorithme

t1 à t6 correspondent aux horaires d’arrivée aux différents points de AB

t7 à t9 correspondent aux horaires d’arrivée aux différents points de AB, ces horaires tiennent compte des durées de correspondance aux transitions.

Les étapes 1, 3 et 5 exécutent des appels RI non détaillée (optimisée pour l’étape 5) en *départ à*.

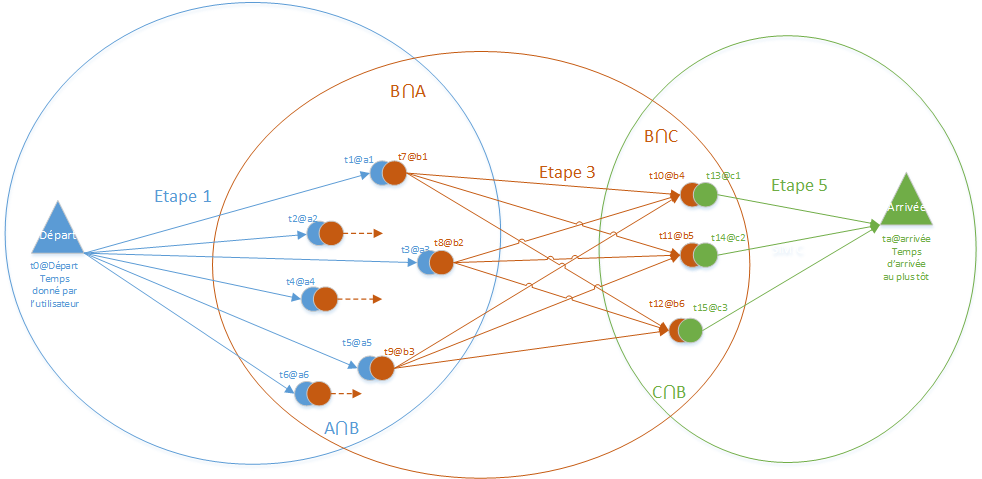


Figure 3 Description géographique des étapes 1, 3 et 5

Le schéma ci-dessous illustre les appels RI pour les étapes 7, 9.

L’étape 7 exécute un appel RI détaillée n-m en *arrivée à* de B⋂A vers B⋂C.

L’étape 8 déduit les horaires de dépar au plus tard de A⋂B par transition

L’étape 9 exécute un appel RI détaillée 1-n en *arrivée à* de loc1 vers A⋂B, le meilleur arrêt a de .A⋂B est identifié

L’étape 10 détermine les horaires sur B⋂a

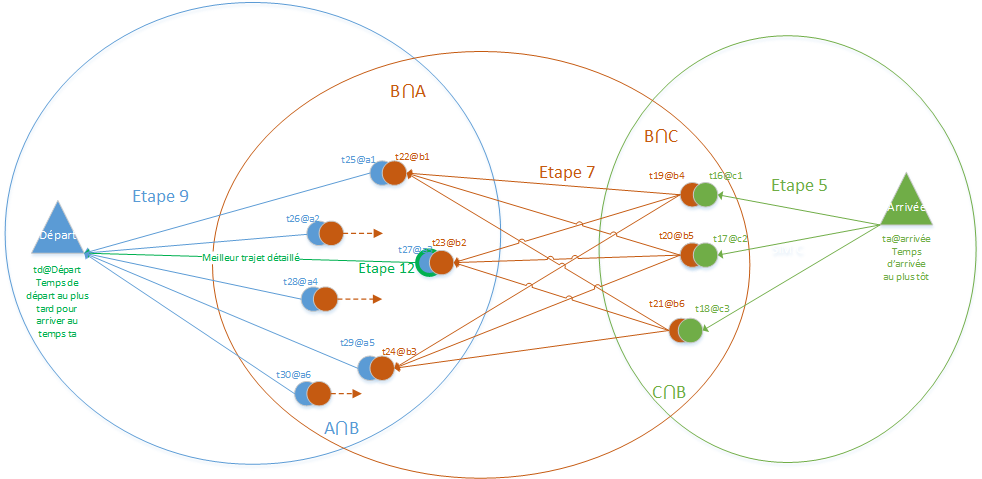


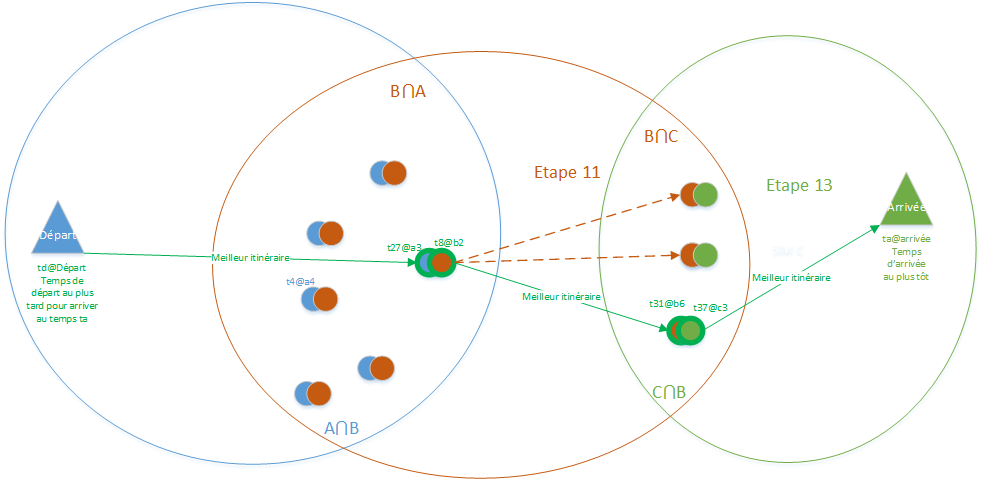
Figure 4 Description géographique des étapes 7, 9

Le schéma ci-dessous illustre les appels RI pour les étapes 11 et 13.

L’étape 11 exécute un appel RI détaillée 1-n en *départ à* de B⋂a, le meilleur point de B⋂a puis de B⋂C sont déterminés.

L’étape 12 trouve les horaires aux meilleurs points de C⋂B par transition.

L’étape 13 exécute un appel RI détaillée 1-n en *départ à* des meilleurs points de C⋂B, le meilleur point de C⋂B est déterminé



##### Détail de l’algorithme de la fonction pour un appel en « arrivée à »

Pour réaliser l’algorithme sur un horaire fourni en « arrivée à » il faut reprendre la même démarche

* En réalisant une permutation sur le nom des ensembles
  + Dans le cas d’une trace (A, B), échanger A et B
  + Dans le cas d’une trace (A, B, C), échanger A et C
* En permutant loc1 et loc2
* En permutant les appels en « départ à » par des appels en « arrivée à » et vice versa

##### Principales caractéristiques de l’algorithme

###### Volume des réponses fournies par les SIM

A chaque fois que l’aiguilleur réalise un *appel «****RI détaillée 1-n*** » celui-ci ne renvoie le détail que d’une seule réponse. Le détail d’une réponse de RI peut être en effet effectivement volumineux.

###### Complexité et nombre d’appels aux interfaces des SIM

La complexité de l’algorithme peut se rapporter au nombre d’appels aux interfaces des différents SIM. C’est en effet les SIM qui portent l’essentiel de la complexité du traitement global.

En partant du principe que tous les SIM sont capables de traiter les appels RI non détaillée n-m, on peut définir les suites suivantes qui comptabilisent le nombre d’appels aux différents services de l’interface générique des SIM pour résoudre une RI basée sur une trace de taille i

* ai la suite des nombres d’appels « RI non détaillée n-m » vers les SIM
* bi la suite des nombres d’appels « RI détaillée 1-n » vers les SIM
* ci la suite des nombres d’appels « RI non détaillée optimisée 1-n » vers les SIM

Pour une trace faisant intervenir 2 SIM (i=2), nous avons donc

* a2 = 1, b2 = 2, c2 = 1

Pour une taille de trace à n+1

* an+1 = an + 2, bn+1 = bn + 1, cn+1 = cn

Il s’agit de suites arithmétiques :

* an = 2\*(n-2) + 1
* bn = n, cn = 1

Dans le cas particulier des SIM qui ont des temps de réponse équivalents pour les 3 services des interfaces génériques, le cout global de la fonction de RI pour une trace de n de SIM correspond à la somme des couts des appels aux interfaces génériques de SIM soit

Cout = an + bn + cn = 3n – 2

Autrement dit, si chaque temps de réponse des 3 services de l’interface générique est de 1 seconde chacun, le temps de réponse de la fonction RI pour une trace de n SIM est nécessairement supérieur à 3n – 2 seconde.

Dans le cas particulier des SIM qui ont des temps de réponse

* de 1 seconde pour « l’Appel RI non détaillée n-m » et « l’Appel RI détaillée 1-n »
* de 2 secondes pour « l’Appel RI détaillée optimisée 1-n »

le temps de réponse de la fonction RI pour une trace de n de SIM est nécessairement supérieur à 3n – 1 seconde

Dans le cas d’un SIM qui n’est pas capable de traiter les appels « RI non détaillée n-m », la place de ce SIM dans la trace a beaucoup d’importance.

En effet, le premier et le dernier SIM de la trace reçoivent des appels « RI non détaillée n-m » avec n=1.

Pour cette raison, la fonction de « Sélection des traces pour une recherche d’itinéraire »

* ne propose que des SIM capables de traiter des requêtes « RI non détaillée n-m » entre le premier et le dernier SIM
* propose dans un trace au maximum un seul SIM non capable de traiter des requêtes « RI non détaillée n-m »

###### Qualité de l’algorithme et critère d’optimisation

L’ensemble de l’algorithme est conçu pour trouver l’itinéraire permettant d’arriver au plus tôt à destination en respectant les autres critères (point de départ, modes, départ à partir de l’horaire fourni).

L’algorithme est construit sur ce critère d’optimisation et garantit qu’il n’y a pas de meilleure solution, ou autrement dit que tous les autres itinéraires possibles (compte tenu des critères) arrivent après celui qui est proposé.

A l’instar de Delfi, c’est ce qui fait la qualité du résultat proposé.

Par contre, il est possible de dérouler ce même algorithme en changeant le critère de sélection du meilleur itinéraire lors des étapes 11, 13 et 15.

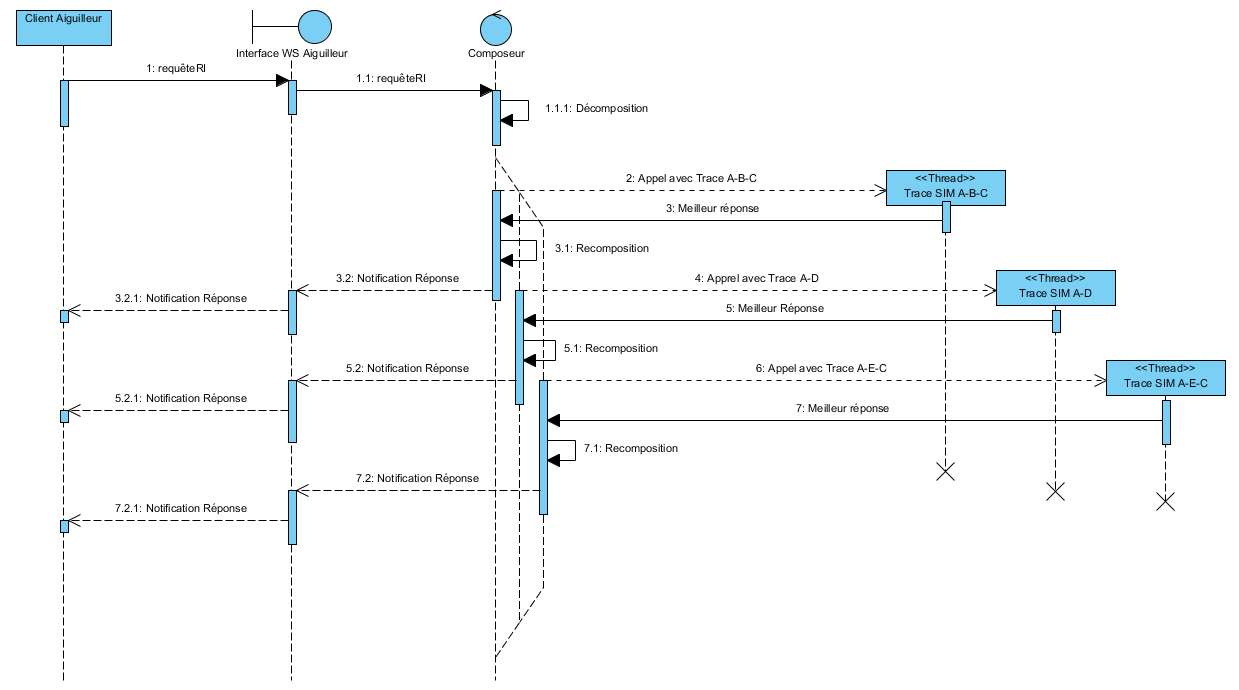
Le critère pourrait être le moins de correspondance par exemple.

Cela dit, le résultat final que propose l’algorithme dans ce cas ne garantit pas que ce soit le meilleur itinéraire possible par rapport au critère proposé.

#### Traitement en parallèle des calculs de RI sur les différentes traces données

La fonction de « distribution des calculs de RI pour une trace donnée » peut être traitée en parallèle pour chacune des traces que le composeur a déterminé.

Le diagramme de séquence ci-dessous illustre comment le composeur peut ainsi notifier le client sans attendre que se termine la totalité des calculs.



#### Optimisation possible par mutualisation de requêtes

Lorsque plusieurs traces partent du même SIM, il est possible de mutualiser la première requête non détaillée.

Pour deux traces A-B et A-C, plutôt que de lancer deux appels non détaillé distinct 1-n en départ à de loc1 vers A⋂B et de loc1 vers A⋂C au SIM A, on peut mutualiser ces appels et lancer un appel de loc1 vers (A⋂B) ∪ (A⋂C), c’est-à-dire vers tous les arrêts de A qui sont en correspondance avec au moins un arrêt de B et tous les arrêts de A qui sont en correspondance avec au moins un arrêt de C. On utilisera alors les résultats dans les étapes suivantes de l’algorithme.

### Phases de développement à venir

Pour entreprendre le développement de l’aiguilleur, il faut prévoir

* de valider le modèle de base des méta-données
* de valider l’interface des services de recherche d’itinéraire et de localités
* de valider les fonctions de recherches d’itinéraires sur l’interface générique

Comme l’aiguilleur est le composant dont la part de R&D la plus importante du POC, il est préférable que le processus de développement reste itératif.

Le développement peut se faire en « bottom up » en commençant par la fonction de recherche de distribution des calculs de RI pour une trace donnée, puis en progressant vers les services de l’interface.

Cela dit, les étapes sont à définir de manière progressive en fonction des difficultés rencontrées.

## Architecture du système d’administration

Le système d’administration met à disposition une IHM qui permet aux administrateurs (cf les acteurs du projet sur le document des spécifications fonctionnelles) de gérer les méta-données.

L’IHM d’administration est en interface avec la base des méta-données uniquement.

Indépendamment de l’IHM, le système gère aussi l’actualisation des tables des arrêts et des transitions par un module d’import.

Pour cela, le module d’import est en interface avec l’ensemble des SIM et la base des méta-données.

C’est la base des méta-données qui permet au module d’import de disposer des informations de connexion aux différents SIM (cf table MIS avec les colonnes APIUrl, APIKey).

### Spécification des principales fonctions

L’IHM d’administration ne présente pas de particularité algorithmique.

Par contre l’actualisation des tables d’arrêts et de transitions nécessite de suivre un algorithme particulier.

#### Fonction de mise à jour des arrêts et des transitions

La fonction n’a pas de paramètre d’entrée ou de sortie.

La fonction appelle dans l’ordre :

* la fonction de mise à jour des arrêts
* la fonction de mise à jour des transitions

#### Fonction de mise à jour des arrêts

La fonction n’a pas de paramètre d’entrée ou de sortie.

La fonction procède de la manière suivante :

1. *Appel des arrêts du SIM* A
2. Détruire dans la table STOPS tous les arrêts du SIM A dont les identifiants n’existent plus. Détruire aussi leurs transitions.
3. Pour tous les arrêts du SIM A déjà existant dans la table STOPS et dont la position géographique a changé :
   * Détruire leurs éventuelles transitions existantes
4. Créer les nouveaux arrêts de A dans la table STOPS
5. Recommencer les étapes 1 à 5 pour tous les autres SIM

#### Fonction de mise à jour des transitions

La fonction n’a pas de paramètre d’entrée ou de sortie.

La fonction utilise plusieurs paramètres

* de distance maximum de transition
* de maximum de transition à un arrêt
* de vitesse nominale de marche
* de vitesse de marche PMR

La fonction utilise les fonctions géographiques et des index géographiques en base.

La fonction procède de la manière suivante :

1. *Pour chacun des STOPS du SIM A n’ayant pas de transition*
2. *Evaluation des arrêts à proximité dans un rayon donné (paramètre) et appartenant à un autre SIM* 
   * Dans la limite du maximum de transitions à un arrêt, les transitions sont crées vers les arrêts trouvés à proximité
   * Les durées de transitions (nominale et PMR) sont évaluées par rapport aux paramètres de vitesse et de distance à vol d’oiseau

Dans le cas où la méta-base est alimentée sur un référentiel d’arrêt partagé, la fonction de mise à jour des transitions doit tenir compte de la hiérarchie des arrêts définie (cf § 4.2.1.4 STOP)

L’index géographique joue un grand rôle. Lorsque la base des méta-données accumule un grand nombre d’arrêts, l’évaluation des arrêts à proximité peut devenir très couteuse sans index géographique.

### Phases de développement à venir

Pour entreprendre le développement du système d’administration, il faut prévoir

* de valider le modèle de base des méta-données
* de valider la fonction d’accès aux arrêts d’un SIM sur l’interface générique

Le processus de développement comporte 2 étapes

* Une première étape pour développer la fonction de mise à jour globale des arrêts sur les SIM
* Une 2° étape pour les IHM

## Architecture du site WEB de démonstration

L’IHM du site de démonstration se résume essentiellement à un formulaire de recherche d’itinéraire et à une gestion de l’affichage des réponses.

### Le formulaire de requête

Le formulaire présente une sélection simplifiée de modes. Cette liste est limitée aux modes des SIM connus sur le POC (cf 5.1.3.2.2.1 Distinction entre les modes définis sur l’interface et le service à l’utilisateur final).

L’IHM propose également un rabattement si les modes associés sont sélectionnés.

C’est l’IHM qui fait la correspondance entre les modes affichés et ceux de l’interface d’échange.

Au niveau des champs de saisie des points de départ et d’arrivée, le formulaire utilise un service externe de résolution d’adresse.

En effet l’aiguilleur n’implémente pas le service de recherche de localités dans le cadre du POC.

A la place, le site WEB s’adresse à un service en ligne tel que Nominatim.

Une fonction de validation permet de s’assurer que le formulaire est suffisamment bien renseigné pour n’envoyer que des requêtes valides.

### L’affichage des résultats de la recherche

Lorsque l’utilisateur valide le formulaire, l’utilisateur reste sur la même page.

La requête de recherche est transmise en tache de fond, le navigateur n’impose donc pas à l’utilisateur un temps d’attente avant d’afficher une nouvelle page.

La page indique néanmoins que la recherche est en cours (le mode d’affichage reste à définir : sablier, animation, message).

Dès la réception de la première notification par l’aiguilleur, l’IHM présente

* la feuille de route reçue,
* indique si d’autre(s) résultat(s) sont attenu(s)
  + L’élément PlanTripNotificationStatus de la structure de réponse fournit les informations nécessaires à ce niveau

Lorsque l’IHM reçoit un nouveau résultat d’itinéraire, celui-ci est intégré à la page:

* une nouvelle feuille de route est accessible
  + la feuille de route peut intégrer une cartographie sous réserve que les SIM aient renseigné des tracés
* l’indication relative à l’attente d’autre(s) résultat(s) est actualisée
  + L’élément PlanTripNotificationStatus de la structure de réponse fournit les informations nécessaires à ce niveau

Le cas échéant, l’IHM prévoit d’afficher

* qu’un incident est survenu sur l’aiguilleur
* que la recherche n’a pu trouver aucune solution et éventuellement le motif (à partir de l’élément PlanTripNotificationStatusCode et de l’élément PlanTripNotificationStatus / Comment )
  + adresse de départ ou d’arrivée trop éloignée
  + date de recherche en-dehors de la période d’offre de transport

### Technologie d’implémentation

Au niveau de l’envoi de la requête et recherche d’itinéraire et de l’affichage des réponses, l’IHM gère en Javascript

* les échanges avec Nominatim
* les échanges en WebSocket avec l’aiguilleur.

### Phases de développement à venir

Pour entreprendre le développement des IHM du site de démonstration, il faut prévoir

* de disposer de la liste exacte des SIM partenaires du projet
* de valider l’interface des services de recherche d’itinéraire de l’aiguilleur et de disposer d’un simulateur de l’aiguilleur accessible en WebSocket

Le processus de développement des IHM du site de démonstration s’organise en 2 étapes :

* une phase de maquettage des pages
* une phase d‘implémentation des interfaces WebSocket .

# Conclusion

L’architecture présentée dans ce document traduit la modularité de la solution proposée.

Le système d’administration et l’aiguilleur ont un couplage minimal qui se limite à la base des méta- données.

Cette forte séparation présente un grand avantage en matière

* de liberté de choix des technologies d’implémentation de ces 2 systèmes
* d’évolutivité des 2 systèmes l’un par rapport à l’autre

Rien d’empêche d’utiliser des frameworks et des langages différents pour développer chacun de ces 2 systèmes.

De même les 2 systèmes qui répondent à des besoins différents, peuvent évoluer de manière indépendante.

Ce document de spécifications techniques marque une étape importante du projet.

Les interfaces au niveau de l’aiguilleur (service de recherche d’itinéraire et localités) et le modèle des méta-données constituent une première contribution au niveau du GT7.2 par rapport à l’élaboration d’un format national de recherche d’itinéraire.

La prochaine étape du projet porte sur la définition des interfaces génériques requises sur les SIM. Un certains nombre d’exigences sur ces interfaces ont déjà été formulées que ce soit

* au niveau technique dans ce document
* au niveau fonctionnel dans le document de spécifications fonctionnelles

Il reste néanmoins à formuler une définition d’interface qui répondent à ces diverses exigences.

La définition de ces interfaces génériques sera l’objet d’une 2° contribution au niveau du GT7.2.

Une fois que ces interfaces génériques seront validées par rapport au cadre de ce projet de recherche, ainsi que celles du service de recherche d’itinéraire de l’aiguilleur, il sera alors possible de produire des jeux de tests.

Les jeux de tests se composent de données de différentes natures :

* un état de la base des méta-données
* une série de requêtes/réponses sur le service de recherche d’itinéraire de l’aiguilleur
* une série requêtes/réponses de SIM

Ainsi constitués, les jeux de test donnent les moyens de décrire différents scénarios

* des traces de 3 ou 2 SIM
* des traces qui comportent un SIM n’ayant pas la capacité à traiter les requêtes n-m
* un SIM qui fournit ou pas un rabattement vélo ou voiture en départ ou arrivée
* des SIM qui fournissent un bilan carbone ou non
* un SIM qui fournit une proposition de TAD
* etc

A partir de ces scénarios

* l’ensemble des réponses attendues de l’aiguilleur doit pouvoir être déterminée de manière unique.
* l’ordre de notification des réponses dépend de la durée d’exécution sur les SIM, une durée plus ou moins longue peut également être simulée

Les jeux de test fournissent les moyens nécessaires à la recette du développement de l’aiguilleur.