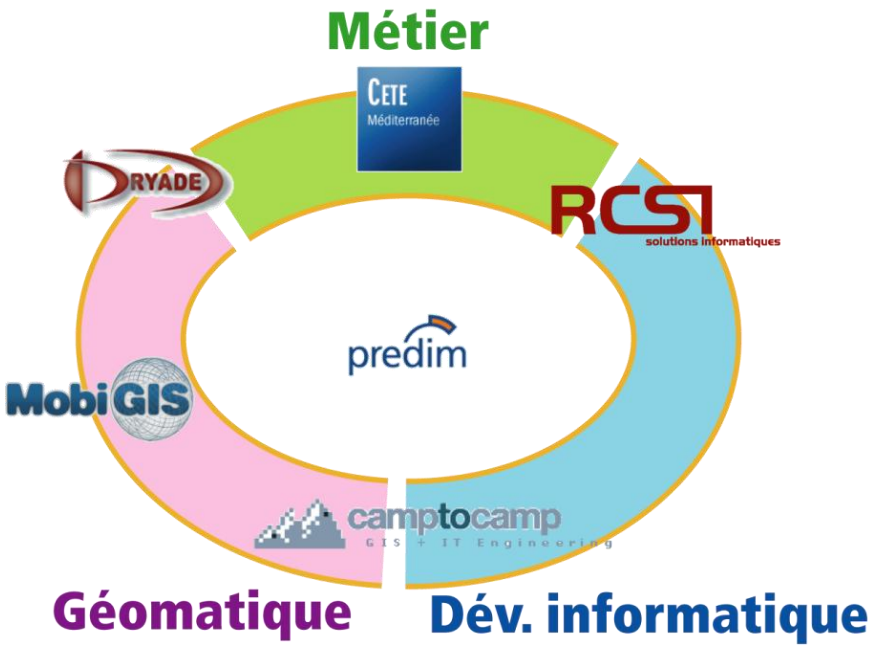


Projet POTIMART

Programmes Open source pour le Traitement de l'Information Multimodale et l'Analyse des Réseaux de Transport

Phase 2 : Dossier de Conception Technique

Mise à jour du 28/11/2008

Version : 8

Partenaires et contacts

Société	Nom du correspondant/Adresse/Téléphone/E-mail
MobiGIS (coordinateur)	Nom du correspondant : Frédéric Schettini Adresse: 27A rue de la Jouclane 31330 Grenade Téléphone : 05 81 60 80 81 E-mail : fschettini@mobigis.fr
CETE Méditerranée	Nom du correspondant : Patrick Gendre Adresse: Avenue Albert Einstein CS 70499, 13593 Aix-en-Provence Cedex 3 Téléphone : 04 42 24 76 87 E-mail : pat.gendre@developpement-durable.gouv.fr
Dryade	Nom du correspondant : Christophe Duquesne Adresse : 17 rue Maryse Bastié 78280 Guyancourt Téléphone : 01 30 21 43 49 E-mail : christophe.duquesne@dryade.net
RCSI	Nom du correspondant : Hugues Romain Adresse : 30 Boulevard Maréchal Leclerc, 31000 Toulouse Téléphone : 05 34 25 33 70 E-mail : hugues.romain@reseaux-conseil.com
CampToCamp	Nom du correspondant : David Jonglez Adresse : Savoie Technolac, BP 352 73377 Le Bourget du Lac Téléphone : 04 79 44 44 96 E-mail : david.jonglez@camptocamp.com

Mises à jour du rapport

N° de version	Date	Auteur principal	Résumé des modifications
0	24/09/2008	F. Schettini (MobiGIS)	Création – Document d'échange avec les partenaires
1	25/09/2008	P. Gendre (CETE Méditerranée)	Révision du chapitre modèle de données
2	Oct 2008	C. Duquesne (Dryade)	Révision du chapitre modèle de données Rapprochement avec les structures Chouette
3	12/11/2008	C.Bloudeau (MobiGIS)	Prise en compte des remarques de Christophe Duquesne et de P. Gendre
4	13/11/2008	P. Gendre (CETE Méditerranée)	Révision du chapitre modèle de données
5	21/11/2008	C. Duquesne (Dryade) et F. Schettini (MobiGIS)	Intégration du chapitre scripting Mise à jour du modèle de données
6	26/11/2008	H. Romain (RCSI)	Chapitre « SYNTHESE »
7	27/11/2008	C.Bloudeau et F. Schettini (MobiGIS)	Chapitre Architecture générale cible: mise à jour du texte Chapitre Modèle de données: révision du texte et ajout d'illustrations Chapitre Scripting: relecture du texte Chapitre Services Potimart: intégration de la contribution de RCSI
8	28/11/2008	P. Gendre (CETE Méditerranée)	Rédaction du chapitre Bilan et poursuite des travaux Consolidation du document

NOTICE ANALYTIQUE

<table>
<tr><td colspan="4">Organisme commanditaire : Ministère de Transports, DGMT</td></tr>
<tr><td colspan="4">Titre : Projet POTIMART</td></tr>
<tr><td colspan="3">Sous-titre : Dossier de Conception Technique</td><td>Langue : Français</td></tr>
<tr><td>Organismes auteurs
Partenaires POTIMART</td><td colspan="2">Rédacteurs ou coordonnateurs
C.Bloudeau, C. Duquesne, P. Gendre, H. Romain, F. Schettini</td><td>date
novembre 2008</td></tr>
<tr><td colspan="4">Résumé :
Ce document constitue les le dossier de conception Technique du projet POTIMART (Programmes Open source pour le Traitement de l'Information Multimodale et l'Analyse des Réseaux de Transport) Phase 2, projet soutenu par la PREDIM en 2007.</td></tr>
<tr><td colspan="2">Mots clés :
Calcul d'itinéraires, information multimodale, analyse spatiale, SIG, logiciel open source, Predim</td><td colspan="2">Diffusion :
Version électronique</td></tr>
<tr><td>Nombre de pages :
47 pages</td><td></td><td>Confidentialité :
Non</td><td>Bibliographie :
Oui</td></tr>
</table>

Sommaire

-

I. CONTEXTE : LE PROJET POTIMART

L'ambition de ce projet soutenu par la Plate-forme de Recherche et d'Expérimentation pour le Développement de l'Information Multimodale (PREDIM www.predim.org) est de développer une suite logicielle sous licence Open Source composée :

- de modules d'import de données pour représenter différents types de réseaux de transport de personnes : Transport Collectifs (TC), Voiture Particulière (VP),
- de modules d'analyse: calcul d'itinéraires TC, VP dans un premier temps, puis à terme de fonctions plus évoluées (accessibilité, etc.),
- d'interfaces (bureautique et internet) Système d'Information géographique (SIG) pour la visualisation des analyses (itinéraires) et des réseaux de transport modélisés.

Le projet comprend deux phases :

- **la première phase réalisée en 2007** a consisté à développer avec des ressources limitées un site Internet de démonstration de certaines fonctionnalités de calcul d'itinéraires multimodal à partir des modules Chouette (données TC), PgRouting (calcul du plus court chemin routier), SYNTHESE (itinéraires TC). Ces modules utilisent les données routières et TC d'un site pilote (agglomération de Toulouse). Ces données sont stockées dans une base de données spatiale PostGIS,

- **la seconde phase réalisée en 2008** permettra de spécifier et si possible de réaliser une application plus complète et documentée sur les bases des besoins du ou des site(s) pilote(s), de la présenter auprès d'utilisateurs potentiels, et de la porter au niveau dans la communauté Open Source SIG.

Le site Internet www.potimart.org fournira au lecteur des informations complémentaires.

II. OBJET DU DOCUMENT

Ce document est le document de conception technique de la plateforme **POTIMART**.

Ce document fait suite au rapport décrivant l'expression des besoins des utilisateurs potentiels (Potimart-Phase2-ExpressionBesoinsCasUtilisation-V4.pdf).

III. STRUCTURE DU DOCUMENT

Le présent rapport comporte les parties suivantes:

- Architecture générale cible
- Modèle de données
- Environnement de scripting
- Services **POTIMART** (PgRouting et Synthèse)
- Bilan et perspectives des travaux

IV. ARCHITECTURE GENERALE CIBLE

L'architecture cible schématisée dans la Figure 1, offrira un ensemble de modules et de fonctions pour répondre à différents besoins de projets (études, services, applications, etc.) dans les thématiques du transport de personnes et de l'information géographique.

La plate-forme **POTIMART** comprendra à terme :

- une base de données SIG Transport permettant de stocker des informations sur les offres de transports. Le modèle de données est défini dans un chapitre suivant et s'appuie autant que possible sur les normes et standards du monde du transport et de l'Open Geospatial Consortium (OGC)
- un environnement de scripting (cf. chapitre VI) pour d'effectuer des analyses « transport » ou des géo traitements simples ou complexes en utilisant des commandes orientés métier
- un environnement de développement modulaire et ouvert qui comprend des services, des bibliothèques d'algorithmes, ou des fonctions d'analyses dédiées au transport (cf. chapitre VII)
- des interfaces d'accès aux données et/ou aux services POTIMART vers des applications clientes : SIG bureautiques, SIG léger (navigateurs Internet), Web Services, etc.

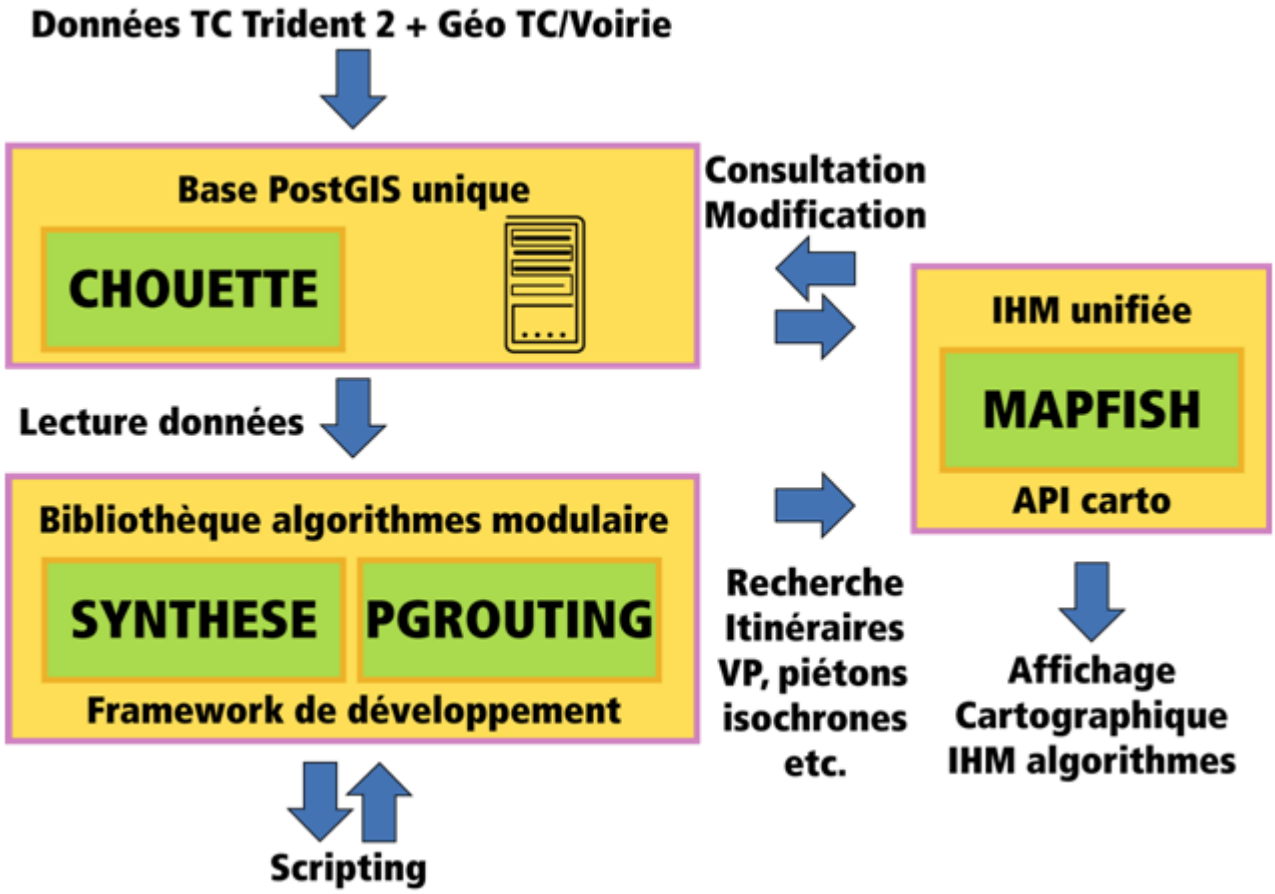


Figure 1: Schéma de l'architecture générale POTIMART

V. MODELE DE DONNEES

Le modèle de données de réseau multimodal défini dans le projet **POTIMART** est constitué de deux éléments majeurs :

- les données constituant le réseau TC,
- et les données représentant un réseau de Voirie.

Les données TC regroupent les lignes de bus et de métro tandis que les données Voirie représentent un réseau de rues et de routes pour permettre le déplacement en mode piéton, voiture ou vélo.

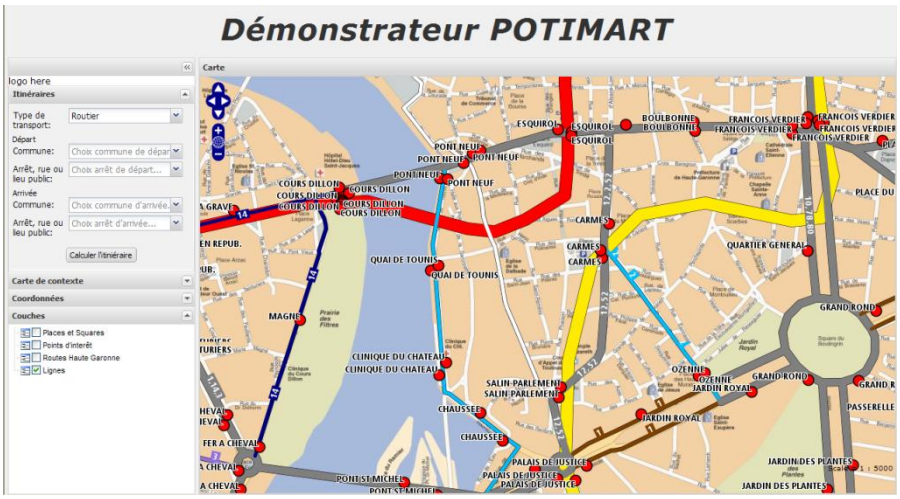


Figure 2: exemple de génération d'une carte représentant des réseaux TC et voiries avec les outils POTIMART

A. Données décrivant le réseau TC

Le modèle de données TC se base sur la structure complète du modèle de données Trident, auquel est ajoutée la dimension géographique, lorsque cela est nécessaire, pour permettre d'effectuer des géo-traitements ou de visualiser certains objets TC dans un SIG.

1. Structure du réseau

Dans le contexte de **POTIMART**, seuls les aspects du modèle Trident liés à la description de la topologie sont pertinents. Les paragraphes ci-dessous présentent les extraits des schémas UML de Trident concernant ces aspects de l'échange (modèle directement inspiré de Transmodel 4).

La Figure 3 présente le modèle générique de description de réseaux : seul le PTNetwork (réseau de TC) est pertinent dans le cadre du projet **POTIMART**.

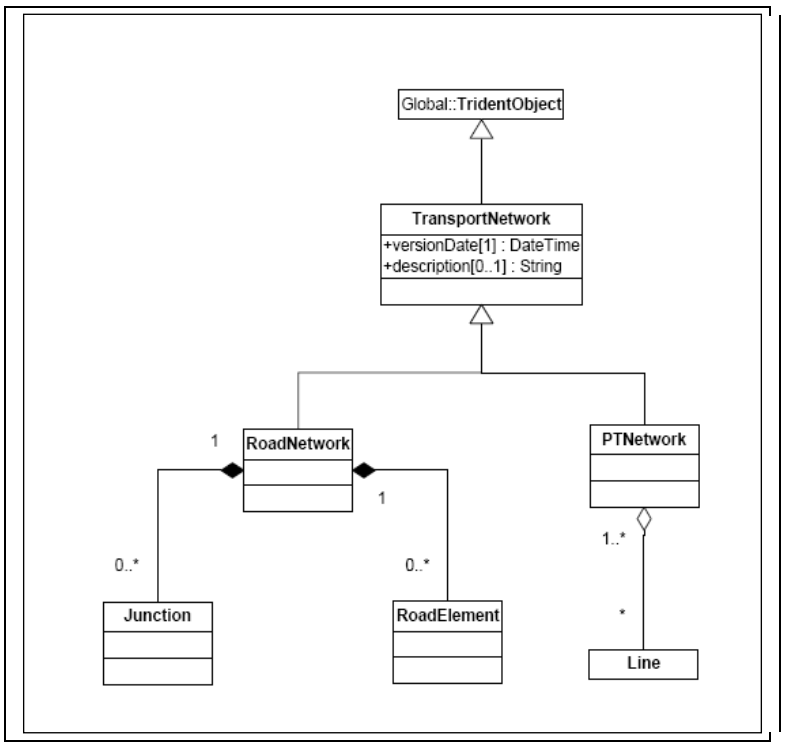


Figure 3: Modèle générique de description de réseaux dans Trident

La Figure 4 suivante présente les principaux éléments de description d'un réseau TC. Parmi les concepts, on distingue les éléments ponctuels (zones d'arrêt), linéaires (lignes TC) et les horaires.

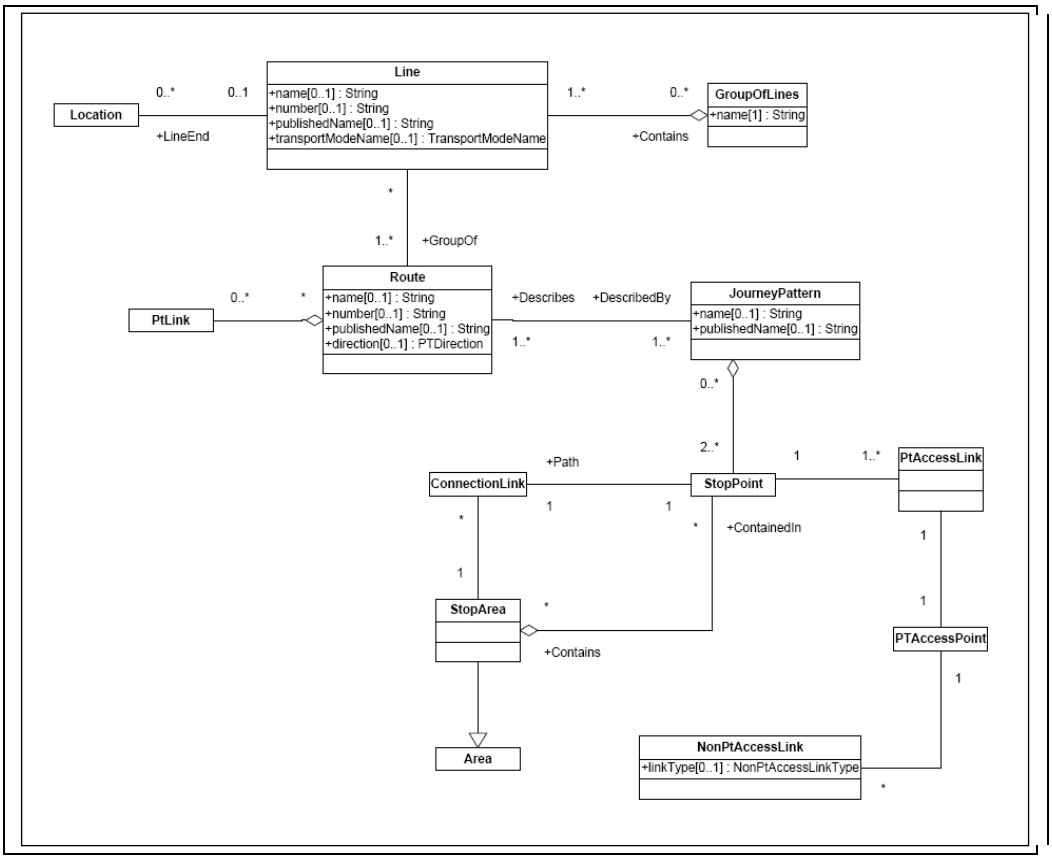


Figure 4: Principaux éléments de description d'un réseau TC dans Trident

Les éléments ponctuels

Les arrêts (StopPoint) constituent l'élément central de la structure Trident. Les StopPoint représentent les arrêts topologiques et ne possèdent pas de coordonnées géographiques.

La notion d'arrêt visible de l'utilisateur (et géo-référencée) est une notion plus agrégée, les StopArea. La nomenclature Trident les définit comme étant la liste des zones d'arrêt Trident qui contiennent au moins un arrêt topologique (StopPoint).

Les StopArea sont de plusieurs types :

- Arrêts Physiques. Ils sont définis comme étant les zones ayant un barycentre sur le poteau
- Arrêts Commerciaux. Il s'agit d'un regroupement d'arrêts physiques, qui sont positionnés au centroïde

Les StopArea (Physique) sont contenus dans les StopArea (Commerciaux).

La Figure 5 ci-dessous fournit une correspondance entre les concepts fondamentaux et les objets Trident.

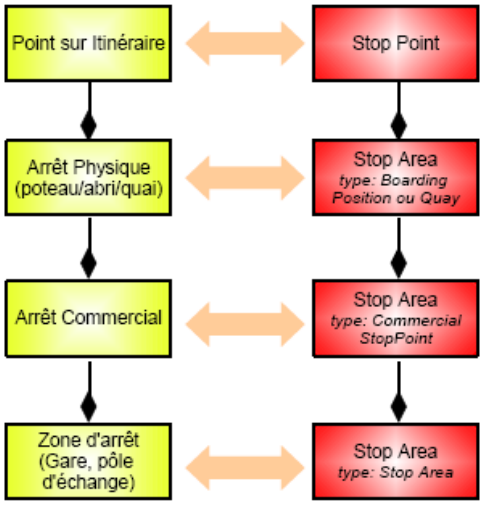


Figure 5: Concepts fondamentaux Trident pour le définiton des zones d'arrêts

Les éléments linéaires

Parmi les principaux concepts, on note :

- une ligne (Line) est constituée d'itinéraires (Route) et peuvent être associées en groupe de lignes (GroupOfLine) pour, par exemple, définir l'ensemble des lignes nocturnes.
- un itinéraire est une séquence d'arrêts ordonnée (donc dans un sens précis aller ou retour), chaque arrêt ayant un unique tronçon (PathLink[1]) prédécesseur (sauf le premier arrêt), et un unique tronçon successeur (sauf le dernier arrêt)
- les itinéraires sont parcourus par des missions (JourneyPattern), correspondant aux arrêts qui seront effectivement marqués (différence entre les omnibus et les directs, mais sur un même itinéraire)
- un tronçon est défini par l'identifiant de l'arrêt par lequel il commence

Les éléments temporels

La Figure 6 suivante définit les horaires sur le réseau TC :

- la course (VehicleJourney) est l'instanciation horaire d'une mission
- les heures des passages aux arrêts pour la course sont présentées par les VehicleJourneyAtStop
- Les courses ont des journées d'application (Timetable) qui peuvent être des jours calendaires (CalendarDay) des types de jour (DayType, pour exprimer les lundis, le week-end, etc.), des périodes (Period) ou encore une combinaison de ces notions

Les notions d'horaire de passage d'un véhicule TC, par exemple un bus, à un arrêt et de date d'application des horaires sont liées à l'objet VehicleJourney de Trident correspondant à la notion de courses. Elles concernent différentes tables nommées Timetable ainsi que la table VehicleJourneyAtStop.

[1] Elément issu de Chouette et non présent dans le modèle Trident

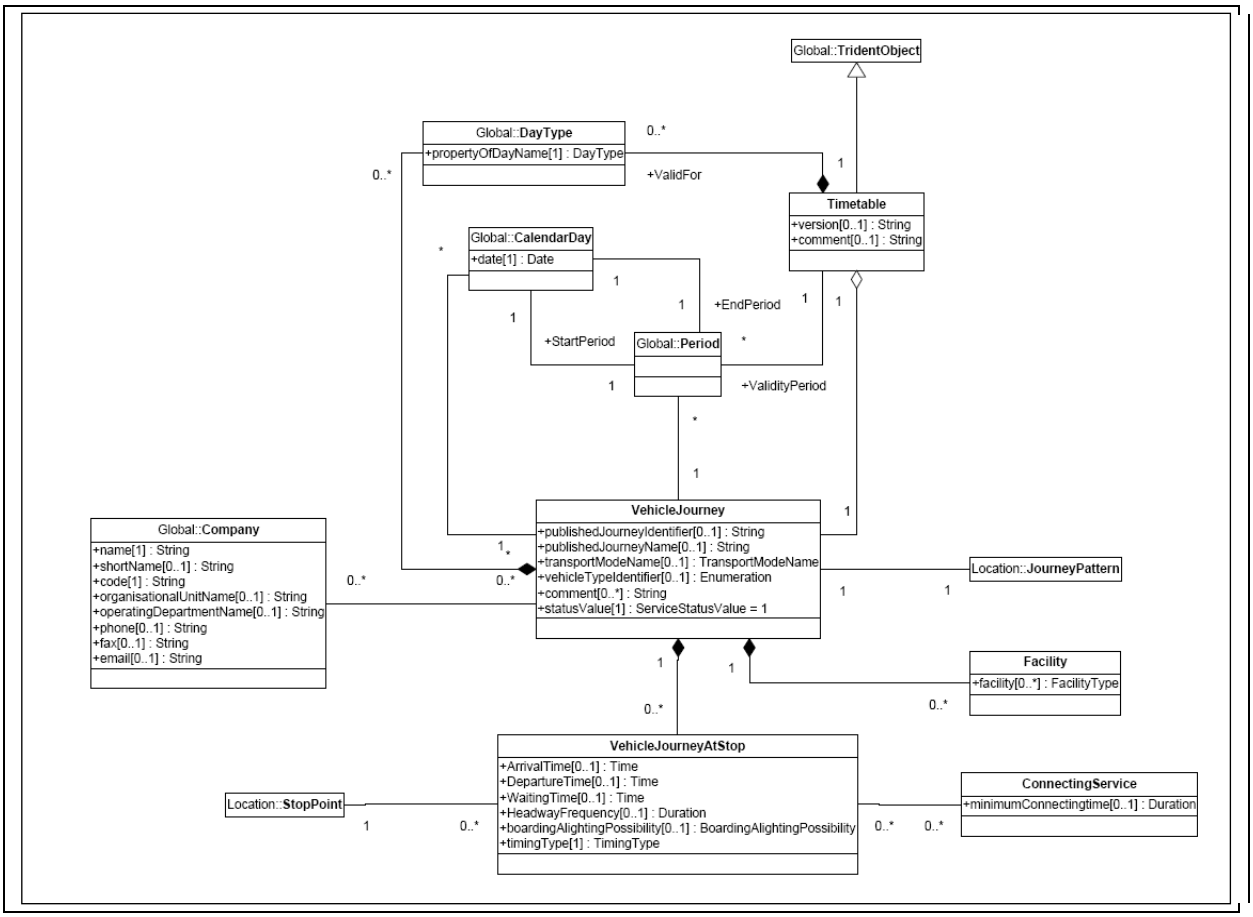


Figure 6: Modèle de données des horaires et claendrier d'application d'un réseau TC dans Trident

2. Extension du modèle Trident pour POTIMART

Dans le modèle TC de **POTIMART**, quatre objets sont étendus pour intégrer une composante géographique :

- StopArea de type Quay ou BoardingPosition (arrêts physiques)
- StopArea de type Commercial StopPoint (arrêts commerciaux)
- Pathlink (tronçons inter-arrêts)
- ConnectionLink (tronçons qui permettent de relier deux lignes différentes)

A noter la nuance entre les notions PtLink de Trident et PathLink de **POTIMART**:

- PtLink : lien inter StopPoints (arrêts topologiques)
- PathLink : lien inter StopArea (arrêts physiques)

3. StopArea

Dans le modèle Trident, la table StopArea regroupe deux types d'arrêts : les arrêts physiques (de type Quay ou BoardingPosition) et les arrêts commerciaux (Commercial StopPoint).

Dans le cadre du projet **POTIMART**, les StopArea de type Quay ou BoardingPosition nous intéressent tout particulièrement. Se sont eux qui vont permettre de définir les StopPoints consécutifs constituant les itinéraires d'une ligne. Ces StopArea comprennent aussi bien les arrêts de bus que les stations de métro.

Pour être géolocalisés, ces StopArea doivent avoir deux champs Latitude et Longitude renseignés. Ces coordonnées permettront alors la création, dans PostgresSQL/PostGIS, d'un champ géométrie rendant possible la création géographique des points.

Tableau 1: Attributs de la couche StopArea (type Quay)

Nom	Type de données	Commentaires
id	bigint	
idparent	bigint	Identifiant du StopArea commercial qui le contient
objectid	character varying(255)	
objectversion	integer	
creationtime	timestamp without time zone	
creatorid	character varying(255)	
name	character varying(255)	
areatype	character varying(255)	
comment	character varying(255)	
registrationnumber	character varying(255)	
nearestPOIname	character varying(255)	
farecode	integer	
longitude	numeric(19,16)	Champ utilisé pour la géolocalisation (WGS84)
latitude	numeric(19,16)	Champ utilisé pour la géolocalisation (WGS84)
longlattype	character varying(255)	
x	numeric(19,2)	Coordonnées X (Lambert II étendu) - non utilisé
y	numeric(19,2)	Coordonnées Y (Lambert II étendu) - non utilisé
projectiontype	character varying(255)	
geom	geometry	Extension POTIMART de type point. Les coordonnées contenues ici sont déduites de celles contenues dans longitude/latitude mais ont un typage 'point' qui permettra une connexion aisée à un SIG. Synchronisation via Trigger
countrycode	character varying(255)	
streetname	character varying(255)	Nom du tronçon de voirie auquel est rattaché le StopArea

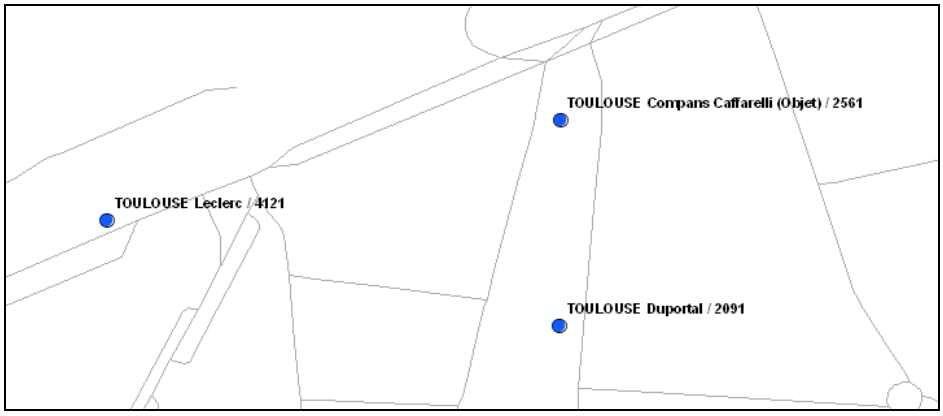


Figure 7: Visualisation cartographique de StopArea de type Quay sur un réseau routier

Les **StopArea de type Commercial** sont également représentés afin de pouvoir les visualiser géographiquement. Comme ils n'ont pas de coordonnées géographiques, il est nécessaire de calculer le centroïde des StopArea Quay qui lui correspondent.

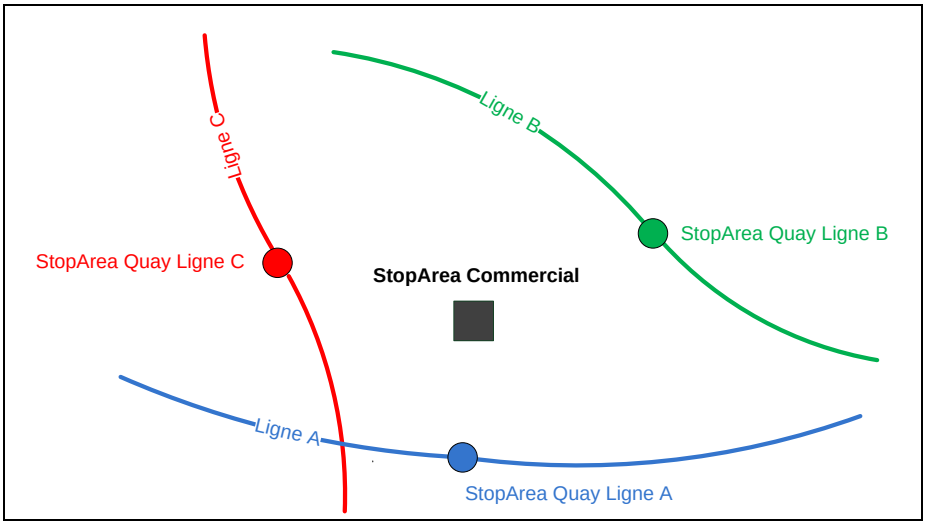


Figure 8: Représentation schématique d'un StopArea type Commercial et des StopArea type Quay

4. PathLinks

Les tronçons PathLink relient les arrêts TC de type Quay ou BoardingPosition. Un PathLink joignant deux points peut-être partagé par plusieurs missions (et donc plusieurs lignes).

En toute théorie, ces tronçons devraient être établis entre des points sur itinéraire (donc des StopPoint pour Chouette et non des StopArea) de façon à permettre que d'un itinéraire à l'autre, un trajet entre deux arrêts consécutif puisse être différent.

Dans un souci de représentation cartographique, ces PathLink devront être éfinis géographiquement. Pour cela, le champ géométrie (geom) est ajouté dans la table permettant de tracer les tronçons en respectant l'ordre de succession des arrêts sur la mission.

Le cheminement inter-arrêt est défini entre deux arrêts consécutifs sur une mission.

Tableau 2: Attributs de la couche PathLinks

Nom	Type de données	Commentaires
id	bigint	
idstartarea	bigint	Identifiant de la StopArea de type Quay ou BoardingPosition commençant le segment
idarrivalarea	bigint	Identifiant de la StopArea de type Quay ou BoardingPosition terminant le segment
Length	bigint	Longueur du cheminement (peut être calculé à partir du champ Geom).
traveltime	real	Temps de parcours : ATTENTION, ce temps n'est pas forcément le même pour tous les véhicules….
geom	geometry	Extension POTIMART de type LINE. Décrit le cheminement d'un véhicule entre deux arrêts (tracé à vol d'oiseau dans un premier temps).

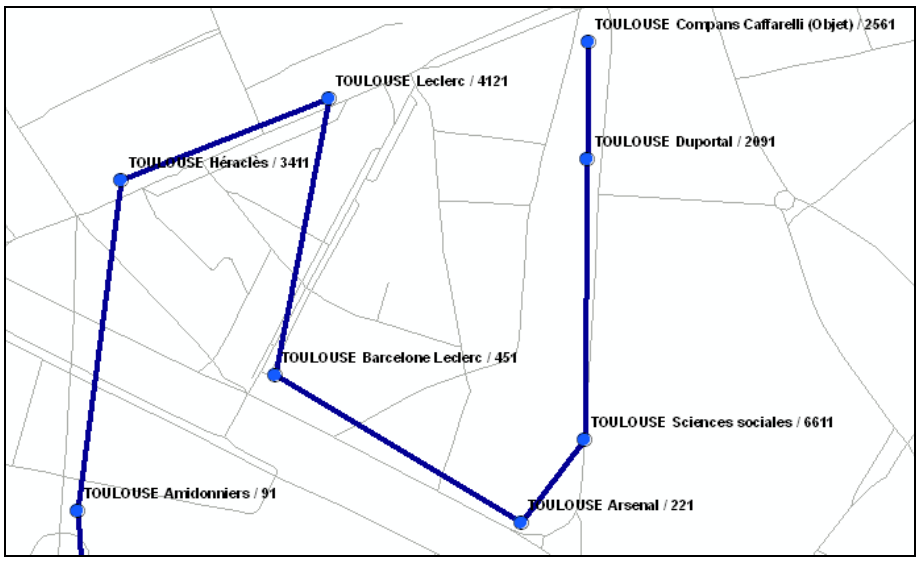


Figure 9: Visualisation cartographique de PathLink et de StopArea sur un réseau

5. ConnectionLink

La table ConnectionLink permet de connecter deux lignes TC différentes sur un même arrêt commercial.

Chaque enregistrement de cette table doit posséder obligatoirement les informations suivantes : un identifiant unique, l'identifiant du StopArea de départ, l'identifiant du StopArea d'arrivée, une longueur et un temps de parcours.

D'autres informations peuvent apparaître comme nécessaires pour l'efficacité du projet Potimart. Toutes ces informations sont regroupées dans le tableau suivant.

Tableau 3: Attributs de la couche ConnectionLink

Nom	Type de données	Commentaires
id	bigint	
id StartArea	bigint	Identifiant du StopArea de départ
id ArrivalArea	bigint	Identifiant du StopArea d'arrivée
objectid	character varying(255)	
objectversion	integer	
creationtime	timestamp without time zone	
creatorid	character varying(255)	
name	character varying(255)	
comment	character varying(255)	
linkdistance	numeric(19,2)	Longueur du tronçon
linktype	character varying(255)	
defaultduration	time without time zone	Temps de parcours par défaut
frequenttravellerduration	time without time zone	
occasionaltravellerduration	time without time zone	
mobilityrestrictedtravellerduration	time without time zone	
mobilityrestrictedsuitability	boolean	
stairsavailability	boolean	
liftavailability	boolean	
geom	geometry	Extension POTIMART de type LINE. Décrit le cheminement de correspondance entre deux arrêts (à vol d'oiseau).

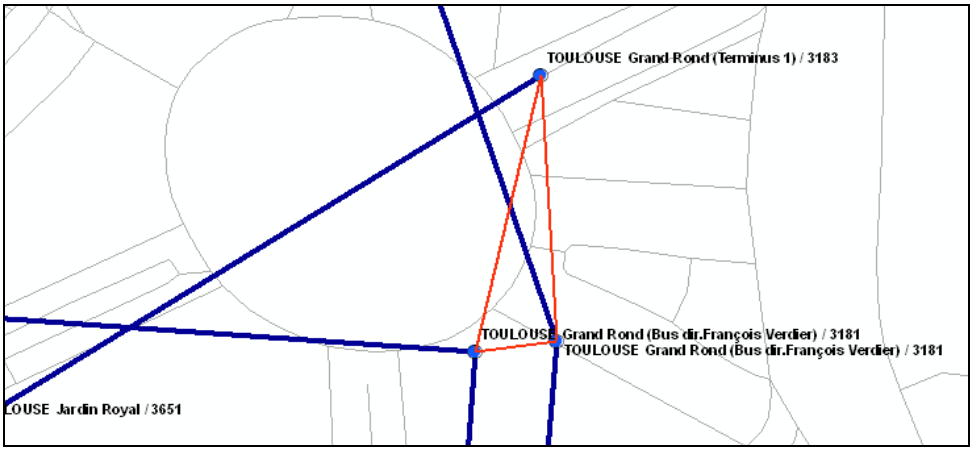


Figure 10: Visualisation cartographique de Pathlink et de ConnectionLink reliant des StopArea de type Quay

6. La notion d'horaires et de dates d'application

Dans le modèle Trident, les horaires sont gérées dans la table VehicleJourneyAtStop qui est directement liée à la table des courses (VehicleJourney) par l'identifiant de la course.

La date d'application de la course concerne la table TimeTable_Date, liée à la table TimeTable qui est liée à la table TimeTableVehicleJourney, elle-même liée à la table des courses (VehicleJourney). Afin de retrouver la date d'application d'une course, il est nécessaire que toutes ces tables soient présentes dans le modèle POTIMART.

7. La notion de fréquence de passage des lignes

La fréquence concerne les arrêts dont les lignes ne sont plus organisées par horaires mais en fréquences de passage (ex : toutes les minutes, toutes les 5 minutes, etc.). Afin de pouvoir modéliser cette notion, absente de la base Chouette, l'objet TimeSlotType décrit dans le modèle Trident est intégré dans le modèle **POTIMART**.

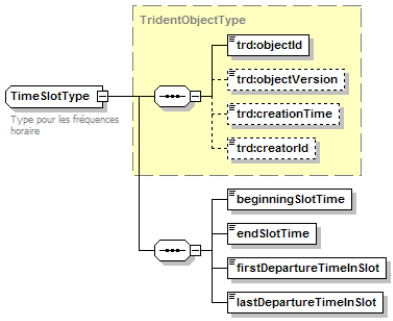


Figure 11: Modèle objet décrivant la notion de fréquence de passage des lignes dans Trident

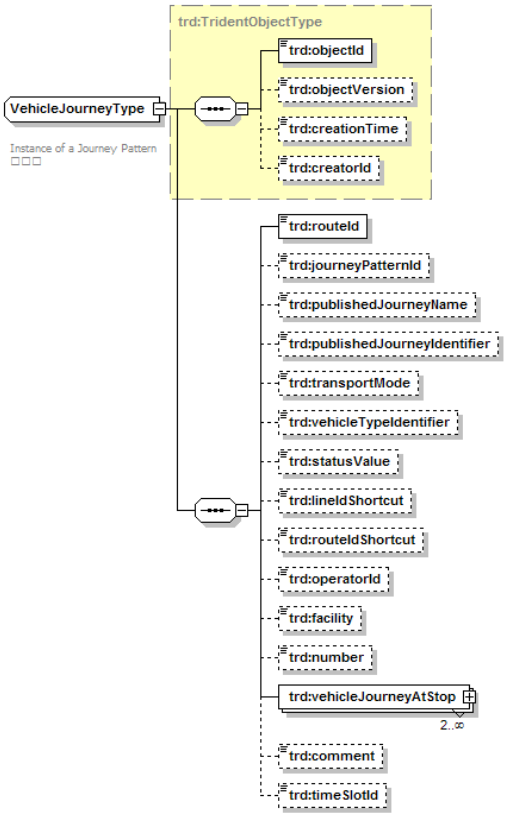


Figure 12: Relation entre l'objet VehiculeJourneyType et l'objet fréquence (TimeSlotType) dans Trident

B. Données décrivant la voirie

1. Streets

La couche Streets du modèle **POTIMART** décrit les données de voirie afin d'effectuer des traitements et des analyses de réseaux pour le mode piéton, voiture ou vélo par exemple. La couche Streets comporte des données attributaires et une composante géographique de type linéaire.

Le Tableau suivant récapitule les attributs nécessaires pour la constitution d'un réseau de voiries de navigation. Parmi les différentes informations, certaines peuvent être considérées comme obligatoires. C'est le cas du sens de circulation, de la longueur du tronçon, ou de la catégorie de vitesse de circulation.

Il est à noter que :

- le format de la table Streets est compatible au format des bases de données commerciales NavStreets-NavTeq et MultiNet-TéléAtlas ainsi que OpenStreetMap. Par conséquent, cette table pourrait facilement être créée à partir de ces sources de données,
- les catégories de vitesse pourront être traitées dans une table externe.

Pour compléter cette table sur la voirie, des couches de données optionnelles et complémentaires peuvent être rajoutées pour enrichir la modélisation :

- les manœuvres interdites ou obligatoires (ex : interdiction de tourner à gauche ou droite, interdictions temporelles de circulation, etc.) pour un mode de transport,
- la description de la voirie routière, intégrant si possible les voies piétonnes, les pistes cyclables, et les voies ferrées
- la description de l'hydrographie
- la description des espaces verts
- la description des zones commerciales
- la description des zones d'intérêt (cimetières, etc.)
- la description des zones « centre ville » (zones de bâti « dense »)
- la description des points d'intérêt (POI)
- les limites administratives et communales

Tableau 4: Attributs de la couche Streets

Champs	Type	Exemple	Commentaire	Obligatoire
IdStreet	bigint	ID 3 ; ID 5	*Identifiant du tronçon*	X
Name	character (255)	Rue Magenta ; Bd Lascrosse	*Nom de la voie*	X
Type	character varying(32)	Rue ; Nationale ; Autoroute	*Type de voie*	X
Numberlanes	integer	2 ; 1 ; 4 ; 6	*Nombre de voies de circulation*	
Directiontravel	integer	1	*Sens de circulation :* • *0 : double sens* • *1 : interdit dans le sens opposé à numérisation* • *2 : interdit dans le sens de numérisation* • *3 : interdit en tous sens*	X
Length	double	154	*Longueur du tronçon (en mètres)*	X
Temps de parcours Piéton	double	10	*En seconde*	
Biketraveltime	double	3	*En seconde*	
Cartraveltime	double	1	*En seconde*	
Speedcat	Integer	1 ; 2 ; 3 ; 8	*Catégorie de vitesse sur le tronçon*	X
Heightrestriction	double	3.8	*En mètre*	
Widthtrestriction	double	6.5	*En mètre*	
Weightrestriction	double	3.5	*En tonne*	
Pedestrianautorize	boolean	Y ; N	*Accessibilité Piéton*	
Publictransportautorize	boolean	Y ; N	*Accessibilité Transports en Commun*	
Carautorize	boolean	Y ; N	*Accessibilité Véhicules Prioritaires*	
Bikeautorize	boolean	Y ; N	*Accessibilité Vélo*	
Truckautorize	boolean	Y ; N	*Accessibilité Camion*	
Taxiautorize	boolean	Y ; N	*Accessibilité Taxi*	
Fromelevation	integer	1	*1 pour niveau supérieur, 0 pour niveau inférieur*	X
Toelevation	integer	0	*1 pour niveau supérieur, 0 pour niveau inférieur*	X
Geom	Geometry		*Type LINE*	X

2. ProjectedPoints

Les arrêts projetés permettent la connexion entre les tronçons de la voirie et les StopArea. Dans cette table, certains attributs sont nécessaires : un identifiant unique, l'identifiant du StopArea auquel l'élément est rattaché ainsi que l'identifiant du tronçon et sa position sur le tronçon pour permettre la connexion avec la voirie.

L'arrêt projeté d'un StopArea peut être représenté en plusieurs points. On peut distinguer :

- le point projeté sur la voirie,
- le point où le bus s'arrête (zébra),
- et le poteau ou l'abri bus.

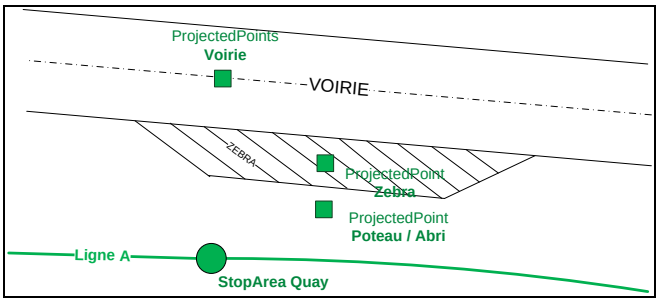


Figure 13: Exemple d'un arrêt projeté (ProjectedPoints) sur la voirie, sur le zébra et au poteau

Par conséquent, les coordonnées géographiques de chacun des points (voirie, zébra, poteau/abri) relatives à un StopArea peuvent être renseignées dans la table ProjectedPoints.

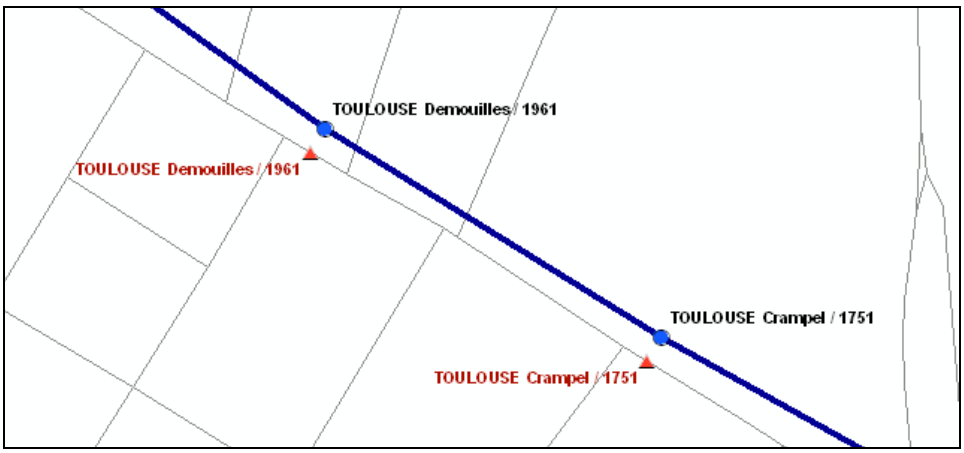


Figure 14: Visualisation cartographique d'arrêts projetés, d'arrêts physiques (StopArea) et de tronçons inter-arrêts (Pathlink)

La table ProjectedPoints

Information	Type	Exemple	Commentaire	Obligatoire
ID	integer	1 ; 2 ; n	*Identifiant unique*	*X*
IDStopArea	integer	2 ; 2 ; n	*Identifiant du StopArea auquel le ProjectedPoint est rattaché*	*X*
Name	character(255)	TOULOUSE Compans Caffarelli	*Nom de l'Arrêt commercial concerné*	
Idstreet	Bigint	1 ; 2 ; n	*Identifiant du tronçon de la couche Streets sur lequel le ProjectedPoint est rattaché*	*X*
Positiononstreet	double	0 à 1	*Position relative projetée depuis le début du tronçon Streets suivant le sens de numérisation*	*X*
Xstreet	double	526767.89	*En Lambert II étendu ou WGS 84*	
Ystreet	double	1844385.69	*En Lambert II étendu ou WGS 85*	
Xzebra	double	526767.89	*En Lambert II étendu ou WGS 84*	
Yzebra	double	1844385.69	*En Lambert II étendu ou WGS 85*	
Xpost	double	526767.89	*En Lambert II étendu ou WGS 84*	
Ypost	double	1844385.69	*En Lambert II étendu ou WGS 84*	
Geom	Geometry		*Type Point dérivé de la position voirie*	*X*

C. Modèle de données multimodal

Le modèle de données unifié permet de créer un réseau dit multimodal, c'est-à-dire interconnectant les éléments d'un réseau TC (modèle Trident étendu) avec ceux du réseau de voiries (couche Streets). La jointure entre les objets des couches StopArea et ProjectedPoints se fait via le champ IDstoparea. La jointure entre les objets des couches ProjectedPoints et Streets peut être attributaire (Idstreets et positiononstreet) ou géographique.
Le diagramme ci-dessous montre les relations entre les objets du modèle.

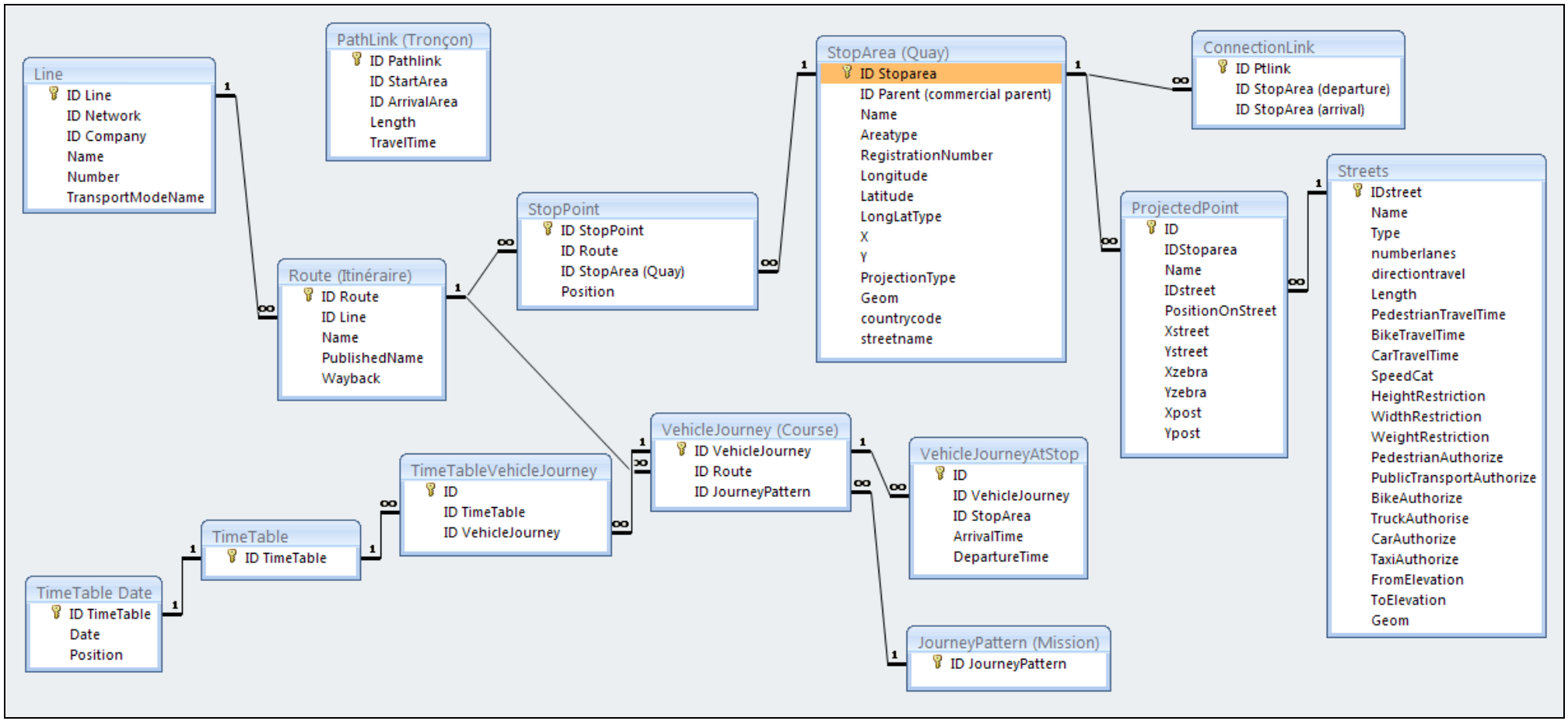


Figure 15: Modèle de données multimodal POTIMART

VI. ENVIRONNEMENT DE SCRIPTING

A. DSL

Le terme Domain Specific Language (DSL) est devenu populaire au cours des dernières années dans le domaine du développement logiciel. Il permet d'identifier un langage de programmation dédié à un problème ou un domaine d'activité particulier.

Le concept n'est pas récent : les langages de programmation conçus pour un type précis d'application et ou de modélisation ont toujours existé, mais le terme est devenu plus populaire, notamment avec l'arrivée de langages comme Ruby qui simplifient la conception de DSL, ou encore avec la récente mise à disposition par Microsoft d'un environnement dédié à la conception de DSL (DSL Tools : http://msdn.microsoft.com/fr-fr/architecture/bb286889.aspx).

À l'inverse on peut considérer les langages génériques comme les langages de programmation comme le C, Java, PHP, etc. qui ne ciblent ni un usage ni un domaine métier particulier.

Comme exemples de DSL déjà largement utilisés on peut entre autres évoquer, YACC pour la création d'analyseurs syntaxiques, Csound, un langage utilisé pour créer des fichiers audio, ou encore GraphViz utilisé pour la mise en page de graphes de tout type, et SQL pour l'interrogation de bases de données.

```
/* Exemple de fichier GraphViz */
/* Une boite CHOUETTE contenant les différents composant interconnectés:
   IHM/Données/IO */

digraph DSLExemple
{
  subgraph cluster0 /* Boite contenant : CHOUETTE */
  {
    label = "CHOUETTE";
    style = filled    ;
    color = lightgrey   ;
    node[style=filled,color=white,shape="box"];

      /* Trois éléments contenus */
    chouette_ihm   [label="IHM"                  ,color=black, fillcolor=white ];
    chouette_data  [label="Gestionnaire de datas",color=black, fillcolor=white ];
    chouette_import[label="Import - Export"      ,color=black, fillcolor=white ];

      /* Connexions entre les éléments */
    chouette_ihm    -> chouette_data;
    chouette_import -> chouette_data;
  }
}
```

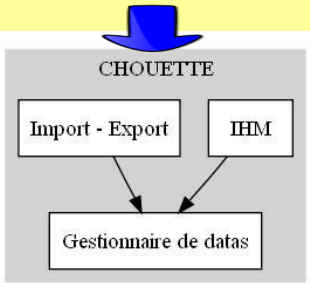


Figure 16 - Exemple de DSL avec GraphViz

Un DSL propose donc un domaine linguistique spécifique, créé pour résoudre les problèmes dans un domaine particulier et ne vise pas à être capable de traiter des problématiques plus génériques (bien que cela puisse être techniquement possible).

Les DSL fournissent un langage (syntaxes ou grammaires), avec des objectifs très spécifiques dans leur conception et leur mise en œuvre. Ils peuvent être visuels (graphiques) ou programmatiques. Par exemple, l'utilitaire de ligne de commande Unix GREP a une syntaxe adaptée à la recherche de lignes de texte. L'outil SED définit lui une syntaxe pour l'appariement et le remplacement basé sur des expressions régulières. Ces petits langages, qui sont fondamentalement de petits DSL, peuvent être utilisés ensemble (dans un Shell) pour effectuer plus de tâches de programmation plus complexes.

Un DSL se situe généralement à mi-chemin entre un petit langage de programmation et un langage de script, et est souvent utilisé d'une manière analogue à une bibliothèque de programmation. Les frontières entre ces notions peuvent sembler assez floues, tout comme la frontière entre les langages de script et des langues génériques.

Si la ligne de démarcation entre DSL et des langages de script est un peu floue, les DSL ne disposent généralement pas de fonctions de bas niveau, pour l'accès au système de fichiers, le contrôle interprocessus, etc. Il est fréquent qu'un DSL ne produise pas de code compilé ou de byte-code interprétable, mais des objets plus complexes : GraphViz produit des données PostScript, GIF, JPEG, etc. Csound produit des fichiers audio, et un DSL de ray-tracing comme POV produit des données graphiques 3D. Un langage informatique comme SQL présente un cas intéressant : il peut être considéré comme un DSL car il est spécifique à la gestion de bases de données relationnelles, mais sa richesse syntaxique et son nombre de fonctions sont plus importants que ceux de nombreux langages de script et, à ce titre, il est souvent considéré comme un langage à part entière.

Il faut aussi noter que de nombreux DSL proposent une Application Programming Interface (API), et peuvent être sollicités à partir d'autres langages de programmation : ils peuvent donc fonctionner comme des bibliothèques de programmation pour un langage générique.

Les DSL sont rendus nécessaires par le besoin de disposer d'un moyen réellement optimisé de traiter un problème. Ils permettent, de plus, à un spécialiste du domaine de rester concentré sur ce domaine sans être obligé d'absorber un grand nombre de nouvelles connaissances informatiques dont beaucoup, en final, ne lui sont d'aucune utilité directe. Un langage généraliste aura tendance à proposer des fonctionnalités qui traiteront certes correctement le problème, mais pas de la manière la plus optimale. Un langage dédié, parce qu'il est conçu à seule fin de traiter un problème ou domaine précis, pourra se délester des lourdeurs inhérentes à une approche généraliste.

Ainsi, les DSL présentent de nombreux avantages : ils utilisent les idiomes et le niveau d'abstraction du domaine ciblé, et sont donc utilisables par des spécialistes du domaine ; ils sont légers, donc facilement maintenables, portables et réutilisables ; ils sont le plus souvent très documentés, cohérents et fiables ; ils sont optimisés pour le domaine. Cependant, utiliser un DSL a un coût : il nécessite de concevoir, implémenter et maintenir son propre langage (ce qui nécessite généralement une double compétence, métier et informatiques), d'éduquer ses utilisateurs, et de définir précisément son champ d'utilisation.

B. Pourquoi un DSL dans POTIMART

Dans le contexte de **POTIMART**, on constate aisément que le périmètre des connaissances nécessaires pour l'utilisation d'un tel environnement est considérable :

- Information sur le transport en commun
- Information sur le domaine routier
- Information et analyse géographique (SIG)
- Cartographie
- Manipulation de graphes complexes
- Recherche d'itinéraire

- Accès au travers d'un SIG de bureau ou au travers d'une interface Web (avec tout ce que cela suppose terme de programmation dès que la moindre personnalisation est nécessaire)

- Tout cela sans oublier les outils sous-jacents : PostGIS, Chouette, Synthèse, MapFish, etc.

Cette richesse constitue de façon évidente un écueil important pour aborder un tel environnement.

L'idée sous-jacente à la mise en place d'un DSL pour **POTIMART** est de proposer à l'utilisateur de se concentrer uniquement sur les domaines métier Transport en Commun et Routier et de lui proposer un environnement de type scripting simple mettant en œuvre le DSL. Dans cet environnement, il n'aura que très peu de choses à apprendre en termes de programmation, et pourra manipuler de façon très directe les objets du domaine métier.

C. Eléments techniques de base

1. Ruby

Ruby a été conçu comme un langage homogène, dont les tenants et aboutissants ont été étudiés avec soins. Son créateur, Yukihiro « matz » Matsumoto, a rassemblé certaines fonctionnalités de ses langages préférés (Perl, Smalltalk, Eiffel, Ada et Lisp) afin d'imaginer un nouveau langage qui mêlerait astucieusement programmations impérative et fonctionnelle.

Il en résulte un langage entièrement orienté objet avec la possibilité de recourir à une syntaxe de programmation procédurale. Les souplesses de Ruby sont adaptées à une large gamme d'applications. La syntaxe du langage est si adaptée à la création de DSL que nombre de librairies Ruby implémentent une interface type DSL.

2. Ruby et les DSL

Pour répondre à une problématique métier en informatique, nous avons la possibilité de construire un langage de programmation spécifique à ce métier, le plus proche possible de celui-ci, pour créer le système d'information le plus précis, flexible et riche possible.

Il est très aisé d'écrire un DSL en Ruby. La part visible d'un DSL n'est souvent qu'une belle interface à un code complexe. On parle souvent de DSL, mais on en utilise encore relativement peu car leur développement sur des domaines complexes est souvent coûteux en temps et en connaissances. En Ruby, les choses sont suffisamment flexibles pour écrire simplement une interface élégante et la plus exacte possible, dans le processus de création du code, dès le départ. Créer une syntaxe métier en même temps que créer la logique métier prend tout son sens en Ruby.

Il est même possible à ce niveau de faire directement participer les clients dans l'écriture du code final et c'est maintenant chose courante en Ruby : nous faisons régulièrement des morceaux de code si clairs qu'un client peut relire et modifier ce code pour modifier le comportement de son application avec des connaissances en programmation quasiment nulle.

3. Outillage disponible

L'outillage pour la création de DSL en Ruby est vaste mais repose sur le fait qu'une fois qu'un programmeur a une connaissance suffisante du Ruby, il peut immédiatement créer des DSL à la syntaxe complexe sans aide d'une quelconque librairie extérieure, s'aidant uniquement des ressources des librairies standard de Ruby.

Comme présenté précédemment, nombre de librairies en Ruby prennent déjà l'apparence d'un DSL et peuvent entrer dans la constitution d'un DSL plus spécifique. Dans le cas de Potimart, c'est l'une des librairies les plus populaires que nous mettons à profit : ActiveRecord, un DSL d'accès à une base de données (autrement dit un ORM, Object-Relational Mapper).

D. Contexte POTIMART

1. Le DSL POTIMART

Le DSL **POTIMART** est donc écrit en Ruby et bénéficie des nombreux travaux récents sur les DSL dans ce langage. Ce DSL peut être découpé en trois parties :

- **DSL Chouette**

Le DSL Chouette est une librairie (écrite en ruby) qui bénéficie d'améliorations constantes depuis un an et demi. Il est basé sur la librairie Ruby ActiveRecord qui lui permet d'accéder aux données Chouette en base de données.

Le DSL Chouette permet non seulement de manipuler avec souplesse les données Chouette, mais il est le support des deux DSL données géographiques et outils d'analyse TC. Cette intégration rend l'utilisation du DSL **POTIMART** encore plus transparente.

- **DSL d'accès aux données géographiques**

Les données géographiques sont stockées en base de données grâce au module PostGIS, qui par ailleurs outille ces données d'une large quantité de fonctions de calcul géographique. Ces calculs sont faits de la manière la plus optimale au sein de PostGIS, mais l'accès à ces fonctions se fait par un DSL haut niveau qui simplifie le mariage de données éparses.

La partie du DSL **POTIMART** travaillant sur les données géographiques PostGIS est totalement intégrée à la partie Chouette. A terme, l'objectif de cette librairie est d'apporter à Ruby l'ensemble des fonctionnalités de PostGIS pertinentes pour le monde des transports. Vu que PostGIS est plus générique que Chouette, ce DSL PostGIS devrait être indépendant du DSL Chouette, voire même, le DSL Chouette devrait d'appuyer sur le DSL PostGIS.

- **DSL d'outils d'analyse TC**

Actuellement, il n'y a qu'un seul outil connecté au DSL, le calcul de graphe de PgRouting. Cet outil a été enrichi (par adjonction de prétraitements et de post-traitements) pour travailler sur des données de réseau de TC (par exemple la création du graphe des missions TC). Mais l'architecture de cette partie du DSL est encore totalement ouverte. De nouveaux développements menés par ailleurs nous dirigent vers une possibilité de mettre sous une même interface, une même syntaxe de DSL, différents moteurs de recherche d'itinéraire (raccordement au moteur Synthèse par exemple).

2. Web - Ruby on Rails

Ruby on Rails a ici le rôle de rendre accessible et aisé d'utilisation du DSL **POTIMART**. Outre donner une interface pour saisir et évaluer du code DSL, cette plate-forme aide à gérer les dépendances du DSL comme, notamment, les bases de données qui requièrent le chargement, le prétraitement, le backup, etc.

E. Implémentation

Dans la Figure 17, on peut aisément distinguer les trois couches du système de DSL et de manipulation de données de **POTIMART**. Au sommet nous remarquons une première couche qui permet de rendre l'ensemble de ces fonctionnalités accessibles sur le web, grâce à Ruby on Rails.

Ensuite dans la couche DSL **POTIMART**, on peut manipuler les données Chouette en CRUD (Create, Read, Update, Delete, c'est-à-dire que l'on peut aussi bien consulter que modifier ces données), on peut accéder aux données géographiques et enfin utiliser les outils d'analyse TC.

C'est dans la dernière couche que tous les calculs et les accès à la base de données se font. Actuellement, nous pouvons remarquer que quasiment tout dépend de la base de données, mais l'architecture du DSL permettra une grande souplesse en matière d'évolution des outils. Ainsi, nous pourrons nous attacher aussi bien à des "Web Services" (services accessible par http, local ou par le web) que directement à des moteurs de calculs en C ou Java par exemple.

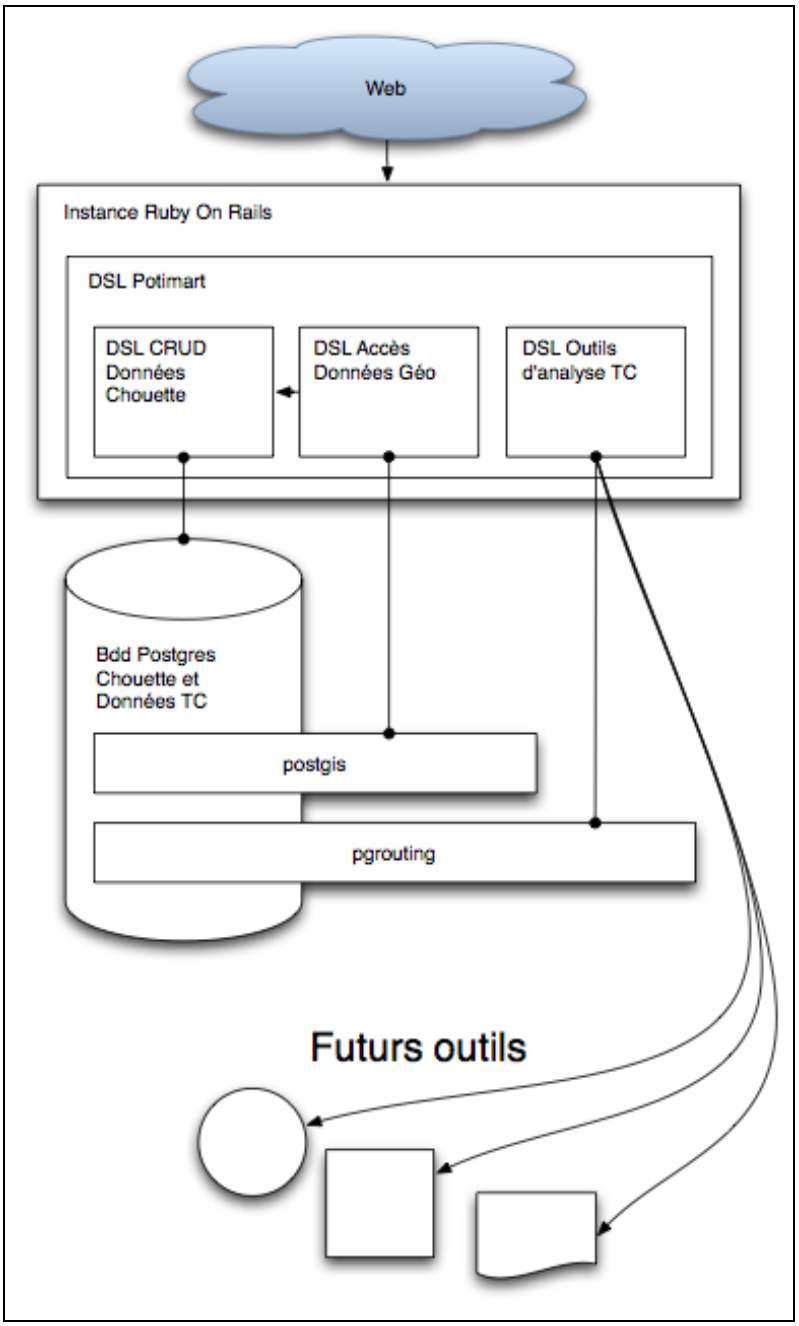


Figure 17: Schéma global d'architecture

1. Interface d'accès

Une interface web développée en Ruby on Rails permet de mettre en action le DSL **POTIMART**. Après le choix d'une base de données avec laquelle travailler, une interface simple et claire permet de saisir des instructions du DSL.

Les réponses sont affichées immédiatement, à côté de la demande. Selon l'information demandée, la réponse sera affichée en conséquence, de la meilleure façon possible.

Cette interface est toutefois indicative, et il sera naturellement possible d'envisager d'autres modes de mise en œuvre du DSL. On peut en particulier envisager de déclencher des scripts DSL à partir d'un SIG de bureau pour déclencher tel ou tel traitement élaboré avec le DSL.

2. Description du DSL (syntaxe et exemples)

La syntaxe du DSL **POTIMART** est relativement simple et autorise souvent plusieurs écritures pour les mêmes tâches. Des similitudes de syntaxe sont souvent recherchées lorsque les fonctions peuvent paraître proches alors qu'elles sont en réalité très différentes.

Voici tout d'abord un lexique simplifié pour l'utilisation de Chouette avec les objets disponibles et leurs propriétés et associations. À savoir les associations retournent d'autres objets décrits ici que l'on peut déterminer d'après leur nom (ex : Company#lines va retourner un array de Line)

- **PTNetwork**
  - propriétés : name, description, sourcename, sourceidentifier, comment
  - associations : lines

- **Company**
  - propriétés : name, phone, fax, email
  - associations : lines

- **Line**
  - propriétés : name, number, publishedname, transportmodename, registrationnumber, comment,
  - associations : company, network, routes, stop_points, stop_areas, unique_stop_areas
  - méthodes : stop_times_on(date)

- **Route**
  - propriétés : idretour, name, publishedname, number, direction, comment, wayback
  - associations : line, stop_points, stop_areas, vehicle_journeys, vehicle_journey_at_stops, journey_patterns
  - méthodes: next_stop_points(stop_point), vehicle_journey_at_stops_from(beginning_stop_area), stop_times_by_hour_from(beginning_stop_area), way

- **StopPoint**
  - propriété : position
  - associations : stop_area, route, vehicle_journey_at_stops, vehicle_journeys, routes
  - méthodes: next_vehicle_journeys(amount?, time?)

- **StopArea**
  - propriétés : name, comment, areatype, longitude, latitude, longlattype, x, y, projectiontype, countrycode, streetname
  - associations : stop_points
  - méthodes: get_commercial_stop_area, complementary_name, zip_code, address, lat, lng, all_stop_points, next_trains(time?)

- **VehicleJourney**
  - propriétés : publishedjourneyname, publishedjourneyidentifier, transportmode, vehicletypeidentifier, statusvalue, facility, number, comment
  - associations : route, journey_pattern, time_table_vehicle_journeys, time_tables, stop_points, vehicle_journey_at_stops

- **VehicleJourneyAtStop**
  - proprieties : arrivaltime, departuretime, waitingtime, connectingserviceid, boardingalightingpossibility, depart
  - associations : stop_point, vehicle_journey

- **JourneyPattern**
  - propriétés : name, comment
  - associations : vehicle_journeys, stop_points, vehicle_journey_at_stops

- **TimeTable**
  - propriétés : date, position
  - associations : time_table_vehicle_journeys, vehicle_journeys

Voici quelques exemples de requêtes sur les données Chouette. Le résultat de chaque opération est indiqué après le dièse, celui-ci étant la clef de commentaire de Ruby et la notation "<#MyKlass>" signifie un objet instancié.

- Accès à une entrée (la première venue) de type "ligne" dans la base Chouette :

```
Line.first # => <#Line>
```

- Accès à toutes entrées d'un type "ligne" dans la base Chouette :

```
Line.all # => [<#Line>, <#Line>, …]
```

- Accès à une entrée d'un type "ligne" dans la base Chouette ayant la propriété "number" égale à 3 (cette technique est valable pour toutes les propriétés de la ligne) :

```
Line.first(:conditions => { :number => 3 }) # => <#Line>
```

- Il existe un raccourci pour retrouver une ligne à partir de son nom :

```
Line["Nom de la ligne"] # => <#Line>
```

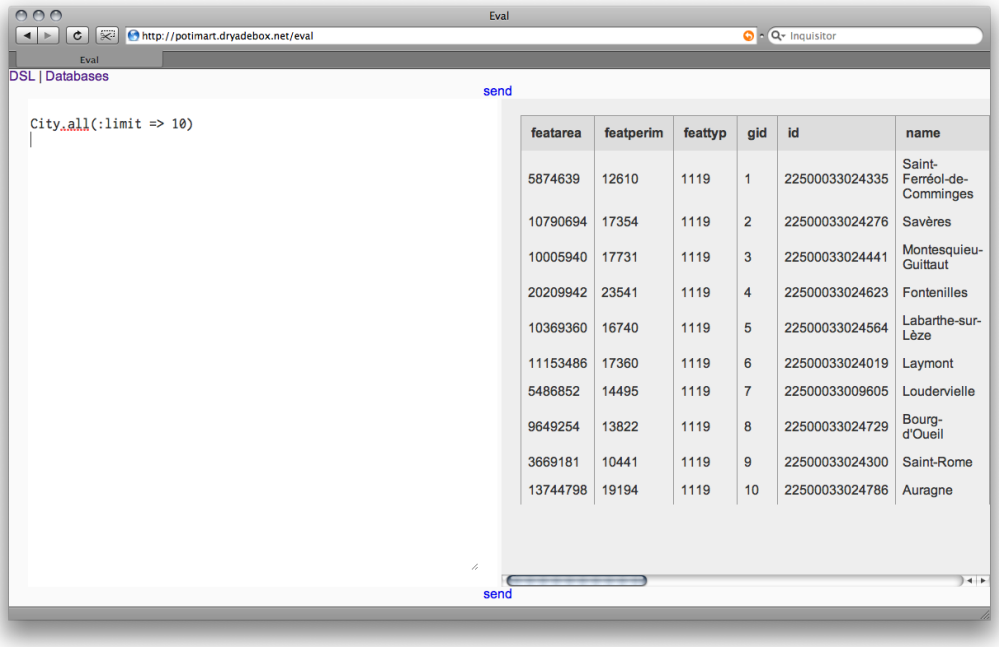
- Modification d'une entrée d'un type "ligne" dans la base Chouette :

```
line = Line.first(:conditions => { :number => 3 }) # => <#Line>

line.number = 4

line.name = "Levallois – Porte de Champerret"

line.save

line.number # => 4
```

- Accès à un POI (point d'intérêt) ou une ville (City) dans la base de données géographique se fait comme une ligne, avec exactement les mêmes possibilités d'éditions :

```
Poi["Nom du point d'intérêt"] # => <#Poi>

City["Nom de la ville"] # => <#City>
```

Vous noterez que le DSL **POTIMART** supporte nativement l'UTF-8 à tous les niveaux.



featarea	featperim	feattyp	gid	id	name
5874639	12610	1119	1	22500033024335	Saint-Ferréol-de-Comminges
10790694	17354	1119	2	22500033024276	Savères
10005940	17731	1119	3	22500033024441	Montesquieu-Guittaut
20209942	23541	1119	4	22500033024623	Fontenilles
10369360	16740	1119	5	22500033024564	Labarthe-sur-Lèze
11153486	17360	1119	6	22500033024019	Laymont
5486852	14495	1119	7	22500033009605	Loudervielle
9649254	13822	1119	8	22500033024729	Bourg-d'Oueil
3669181	10441	1119	9	22500033024300	Saint-Rome
13744798	19194	1119	10	22500033024786	Auragne

Figure 18: Exemple de consultation des 10 premières communes

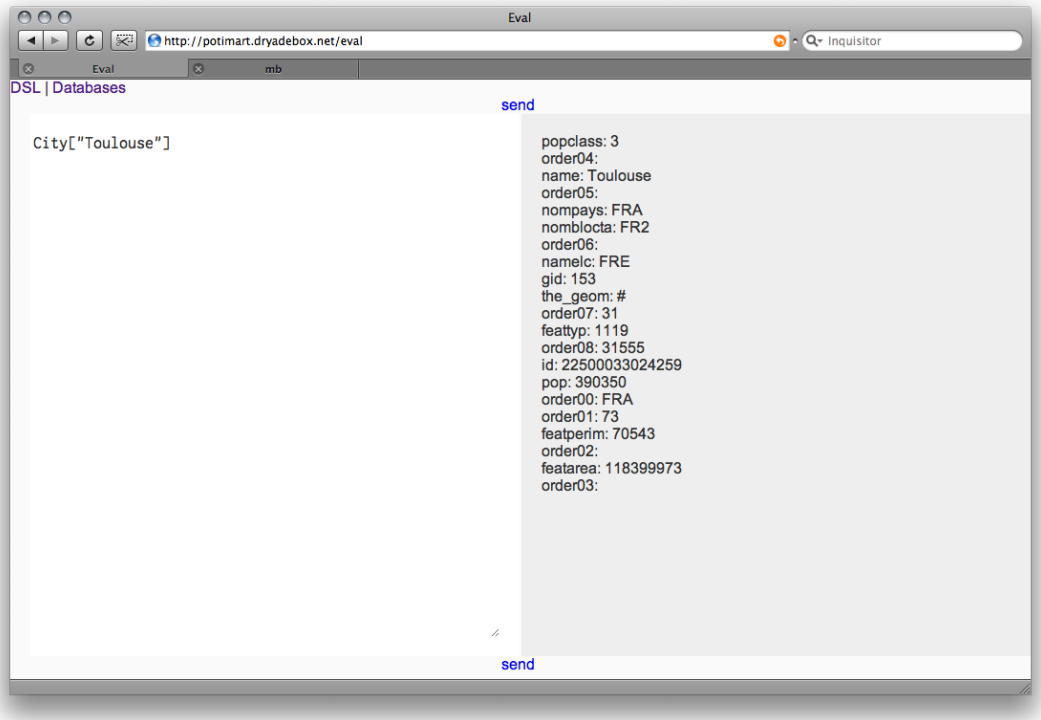


Figure 19: Exemple de consultation des informations sur la ville de Toulouse

- Pour accéder aux objets liés aux premiers une quantité de méthodes permettent de faire une association automatique entre les entrées.

```
Line = Line.first

stop_areas  = line.stop_areas # => [<#StopArea>, <#StopArea>, …]

# Nous passons ici dans chaque object "Station" de la

# première station trouvée et nous affichons le nom

# de la station

stop_areas.each do |stop_area|

  puts stop_area.name

end
```

- La base Chouette est liée ainsi à la base de données géographique. Ainsi, en partant de la ville, on peut obtenir toutes les stations associées explicitement à cette ville.

```
City["Toulouse"].stop_areas # => [<#StopArea>, <#StopArea>, …]
```

- Mais l'on peut également vouloir la même chose mais en ne prenant uniquement compte des positions géographiques des stations et du pourtour de la ville :

```
StopArea[City["Toulouse"].bounds] # => [<#StopArea>, <#StopArea>, …]
```

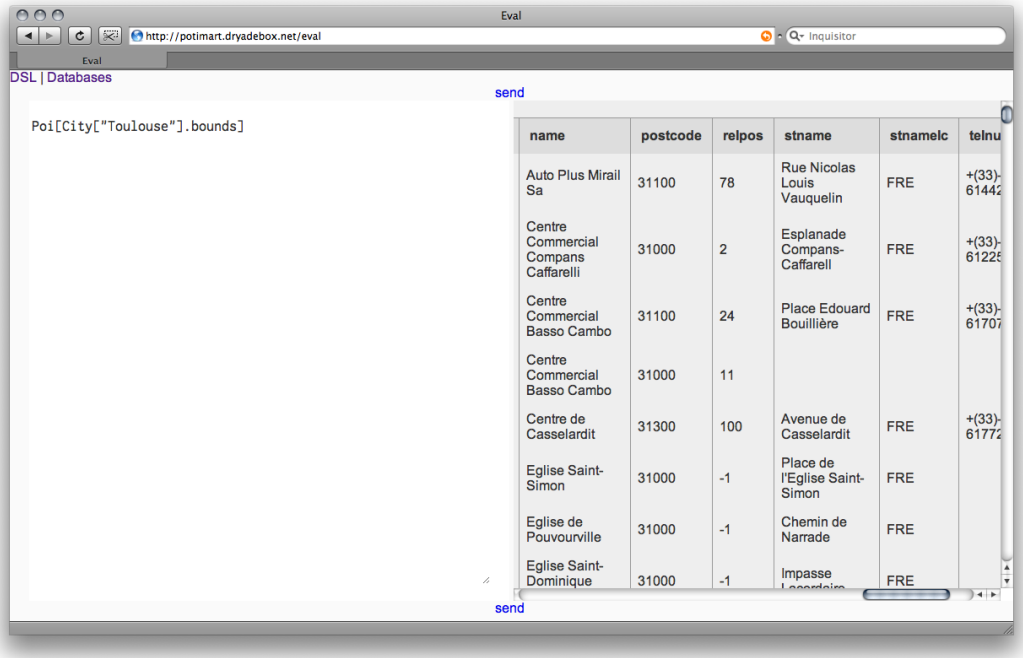


Figure 20: Exemple de consultation des communes jouxtant Toulouse

- PgRouting s'intègre dans cet accès aux données de manière transparente. Ainsi, le moyen le plus simple de faire une recherche d'itinéraire est de simplement spécifier les noms des départ et arrivée souhaités. Actuellement shortest_path ne prend pas d'argument supplémentaire ; à terme il est envisagé de rajouter de nouvelles options à shortest_path tel que les missions, lignes ou stations à éviter.

```
Shortest_path("Les Halles", "Gare du nord")
```

En retour on reçoit une liste d'arrêts et de changements.

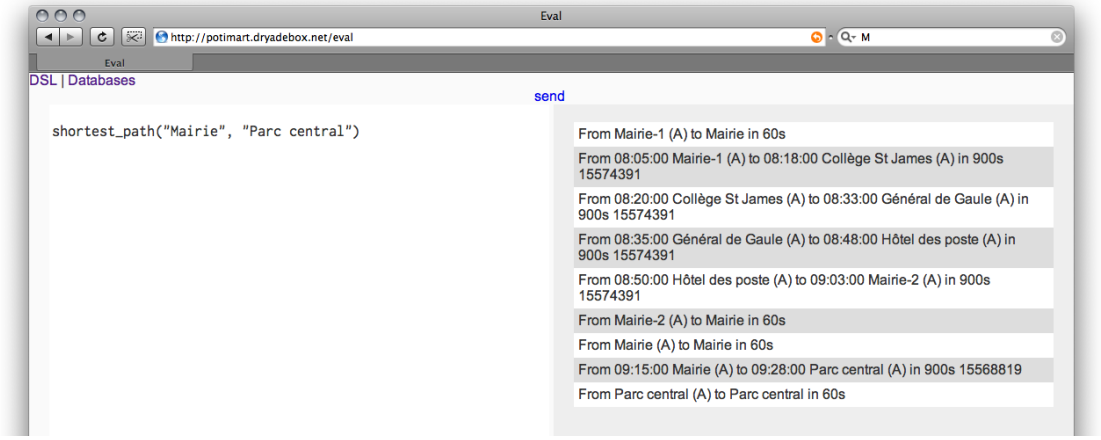


Figure 21: Exemple de recherche d'itinéraire avec PgRouting sur un réseau de test

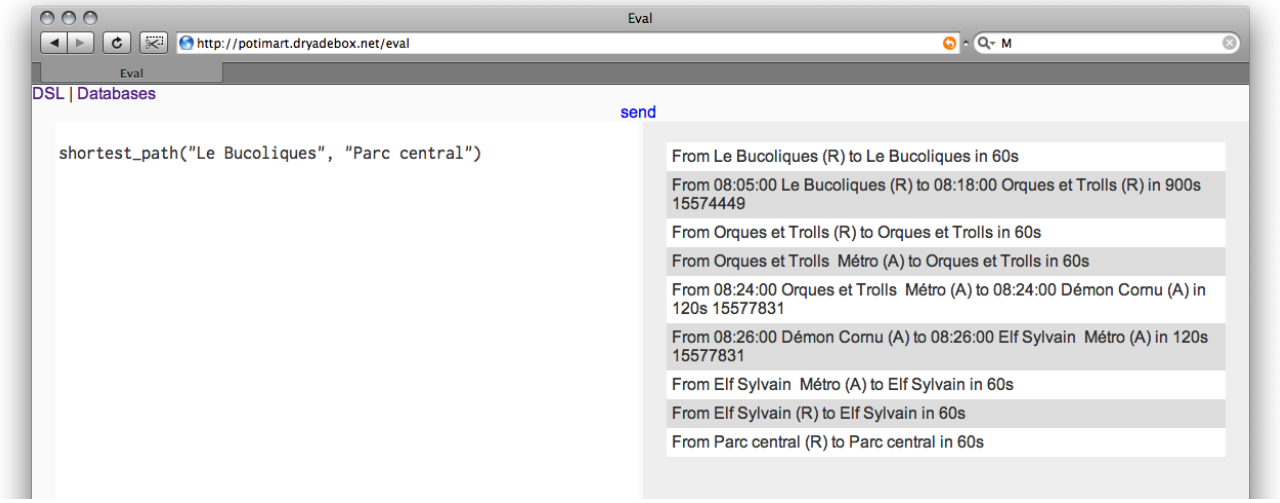


Figure 22: Exemple de recherche d'itinéraire avec PgRouting sur un réseau de test

3. Accès à Chouette

L'accès à Chouette par son DSL est un projet qui a commencé avant le projet **POTIMART** et qui a bénéficié de long mois de maturation. En plus d'un accès direct simple et efficace aux données de la base de données Chouette, nombre d'outils environnent ces données pour exploiter de différentes manières cette base. Ainsi, il est également possible de modifier très aisément les données, créer de nouveau jeu de données en quelques lignes de DSL, extraire des ensemble d'horaires, etc.

4. Utilisation de PgRouting

Principe

PgRouting est un plug-in pour la base de données PostgresSQL pour faire du calcul d'itinéraire type plus court chemin (et ne gérant pas actuellement les contraintes du TC) avec différents algorithmes.

Dans **POTIMART**, un DSL fait l'interface avec PgRouting. L'intégration de ce DSL avec les autres DSL du système rend l'utilisation de PgRouting une tâche très aisée. Certaines tâches de maintenance, telle que la création de jeux de données pour le moteur de recherche, sont intégrées à l'interface web.

Vous pouvez ainsi chercher un arrêt dans Chouette, ou des arrêts proches d'un élément géographique contenu dans le jeu de données géographiques du système, pour le passer à PgRouting et recevoir en retour une réponse structurée sans autre configuration.

Intégration TC

PgRouting s'appuie sur les objets de base pour la description d'un réseau, à savoir les arcs et les nœuds, chaque arc disposant d'une valorisation qui sera utilisée lors du calcul de plus court chemin (le but étant de minimiser le coût total du trajet).

Les réseaux TC ont une particularité forte : les arcs n'ont pas de valeur fixe (comme la distance ou le temps de parcours pour le réseau routier, même si ce dernier peut évoluer dans le temps en fonction du trafic), mais des valeurs liées à un horaire (le bus ou le train ne passe qu'à certaines heures discrètes) et valable pour certains jours uniquement (le bus fonctionnant du lundi au vendredi peut ne pas fonctionner les samedis et dimanches). Or, les algorithmes proposés par PgRouting, Dijkstra ou A*, ne sont absolument pas en mesure de prendre directement en compte une telle spécificité.

Il a donc été nécessaire de mettre en place une solution appropriée.

Son principe consiste à créer un graphe sur la base des missions, des arrêts, des zones d'arrêt et des correspondances :

1. Pour deux arrêts physiques (StopArea de type Quay ou Boarding Position) consécutifs dans une mission, on crée un arc dans PgRouting, valorisé par le temps de parcours moyen des courses qui l'empruntent (dans le cas de l'expérimentation, seule la valorisation d'une course de référence a été utilisée).

2. Les arrêts physiques (StopArea de type Quay ou Boarding Position) sont raccordés, par un arc PgRouting, à l'arrêt commercial auquel ils appartiennent (valorisé par un ConnectionLink s'il est disponible et par une valeur forfaitaire sinon)

3. Les arrêts commerciaux sont raccordés, par un arc PgRouting, à la zone d'arrêt à laquelle ils appartiennent (valorisé par un ConnectionLink s'il est disponible et par une valeur forfaitaire sinon)

4. Les zones d'arrêt connectées entre elles par un Connection Link le sont aussi par un arc PgRouting.

On obtient ainsi un « réseau de missions » qui peut être directement exploité par PgRouting. Un fois le résultat obtenu, on applique les horaires directement lus dans la base Chouette, en utilisant les premiers passages suivant l'heure d'interrogation. Une telle méthode donnera bien comme résultat le trajet le plus rapide, en termes de temps passé dans les transports, mais pourra « laisser passer » des solutions plus longues impliquant une attente (au départ ou en correspondance) plus courte. Il peut aussi arriver que les missions remontées ne proposent aucune course pour le jour d'interrogation : il est alors possible de relancer le calcul d'itinéraire, en supprimant les courses non pertinentes (PgRouting utilise une description de réseau qui lui est transmise par un SELECT, une exclusion par un simple « NOT IN » reste donc très facile à mettre en place).

F. Intégrations futures

1. Principe

L'architecture actuelle de l'ensemble DSL **POTIMART** est entièrement ouverte aux améliorations. Il sera possible d'ajouter de nouvelles fonctionnalités au DSL et à l'interface pour une multitude de possibilités.

Le DSL peut s'enrichir en gardant sa syntaxe actuelle pour supporter davantage de types de données en entrée, d'autres moteurs de recherche d'itinéraire, de nouvelles options de visualisation, etc.

L'interface Web pourra permettre le partage de résultats, la protection des données, un historique de tout le travail effectué, des moyens de communication entre les utilisateurs, etc.

L'ensemble de ces améliorations peuvent prendre deux voies : des améliorations générales de **POTIMART**, qui seront intégrées d'office au cœur du système ; ou des fonctionnalités précises pour certains utilisateurs qui en auront besoin et qui pourront non seulement être intégrées au système sans en modifier le cœur, mais qui pourront être ajoutées à la demande et partagées entre les organismes.

2. Domaine routier

Le prototype d'intégration de PgRouting au DSL a été réalisé sur la base du réseau TC (qui est le cas le plus complexe). Il sera intéressant de pouvoir étendre le réseau au domaine routier.

Pour ce faire, il faudra :

- créer un réseau PgRouting sur la base du réseau routier ce qui ne pose pas de difficulté technique, et qui a déjà été réalisé dans le cadre des démonstrateurs POTIMART lors de la phase 1. Il sera toutefois sûrement utile de prévoir l'extraction de plusieurs réseaux : à minima un pour les parcours routiers, et un pour les parcours piétons. Un réseau dédié aux trajets en vélo peut être envisagé.

- connecter les nœuds du réseau routiers aux nœuds du réseau TC (voir le chapitre Modèle de données multimodal) avec des arcs PgRouting.

Une fois cela fait, et avec un post-traitement adapté, il sera possible de réaliser des recherches d'itinéraires multimodales (en utilisant des variantes du SELECT utilisé par PgRouting, pour n'emprunter que les éléments du réseau souhaités).

VII. SERVICES POTIMART

1. Recherche d'Itinéraires via le module PgRouting

2. Recherche d'itinéraire TC via le module Synthèse

Fonctionnalités générales

La recherche d'itinéraires TC de Synthèse permet, à partir de la base voirie et TC, de calculer l'ensemble des solutions utiles d'un point A à un point B sur une plage horaire déterminée, et à une date donnée.

Le résultat est une grille synthétique présentant chaque solution sous forme de colonne, indiquant les numéros de lignes à emprunter, les trajets à pied, ainsi que les heures de départ, de changements, et d'arrivée, la durée des trajets, les messages conjoncturels (déviations, obligation de réservation, etc.). La grille donne accès à un descriptif littéral de chaque solution, appelé « feuille de route ».

Le format du fichier résultat est paramétré par les données d'interface : il est autant possible de produire plusieurs versions HTML plus ou moins dynamiques, que de fabriquer des sorties XML, binaires, ou selon tout autre format spécifié. Synthèse peut donc s'intégrer aisément dans une chaîne complète de calcul, comme l'environnement de scripting de **POTIMART** par exemple.

Utilisation

Le formulaire d'entrée

Figure 23: Exemple de formulaire d'entrée de Synthèse

- Lieux de départ et d'arrivée

Pour désigner un lieu de départ, taper le nom de la commune dans laquelle se trouve le lieu. Pour valider la commune tapée, il est possible de cliquer sur « Guide » pour faire apparaître une liste de propositions.

Puis taper le nom du lieu public ou du point d'arrêt désiré dans la case située en dessous, en utilisant éventuellement la fonction « Guide » pour valider le choix.

Si aucun point d'arrêt n'est spécifié, le point d'arrêt central de la commune sera désigné automatiquement (ex : Toulouse Matabiau). Ce peut être le cas si une personne demande des informations pour aller dans une commune sans connaître l'arrêt précis.

Faire de même pour désigner le lieu d'arrivée. Puis choisir la journée sur laquelle s'effectuera la recherche.

- Options

Par défaut la période de la journée choisie sera toujours la journée.

Pour désactiver les propositions faisant intervenir des lignes non couvertes par la tarification Tisséo (exemples : les autocars départementaux, les autocars régionaux, les TER…), cocher la case correspondante « Tarification Tisséo uniquement ».

- Valider

Le bouton « entrée » du clavier équivaut à un clic sur « rechercher votre itinéraire ». Pour afficher le retour, sélectionner d'abord la bonne date, puis cliquer sur « itinéraire retour ».

La grille des résultats de Synthèse

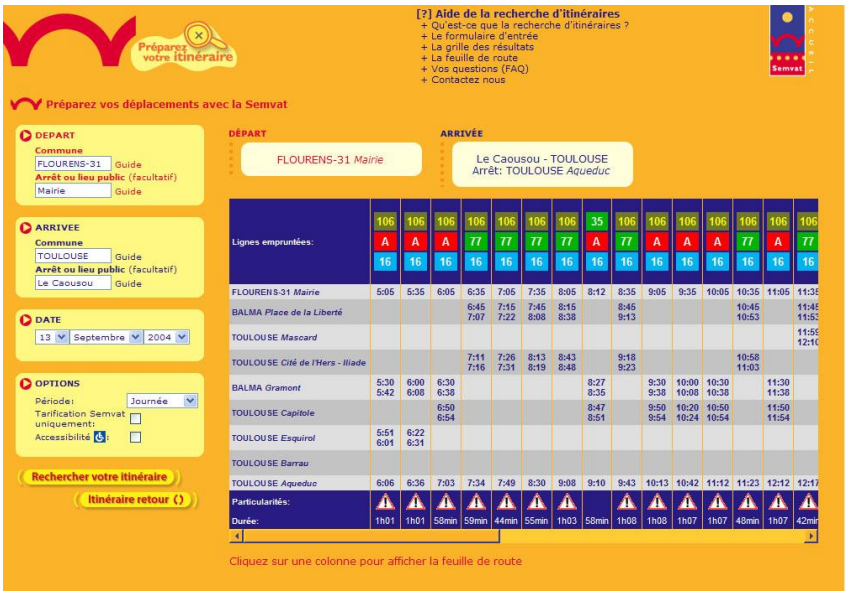


Figure 24: Exemple d'une grille des résultats générée par Synthèse

- Présentation

La grille des résultats permet de voir d'un seul coup d'œil les solutions proposées sur la période de la journée choisie. Chaque colonne représente une solution.

En haut de chaque colonne sont indiqués les numéros des lignes à utiliser. Les heures intermédiaires indiquent les heures des correspondances, en face du nom de l'arrêt de correspondance. Noter que les arrêts de correspondance peuvent différer d'une solution à l'autre.

En bas de chaque colonne est indiquée la durée de parcours maximale.

Cliquer sur une colonne pour obtenir la feuille de route détaillée correspondante.

- Services continus

Dans certains cas, des colonnes se présentent sous forme de plages horaires (Ex : 8 :00 à 12 :00). Ces colonnes indiquent que la solution proposée est valide en continu sur la plage horaire indiquée. Dans ce cas, les temps de parcours affichés incluent l'attente aux arrêts.

- Services continus

Dans certains cas, des colonnes portent un panneau  .

Il indique que des informations complémentaires importantes sont présentes sur la feuille de route. En général, ce panneau s'affiche pour indiquer des lignes à réservation obligatoire ou pour communiquer des informations exceptionnelles (travaux, etc.).

- Marche à pied

Dans certains cas, il est proposé de rejoindre un arrêt à pied pour effectuer une correspondance. Les parties à effectuer à pied sont indiquées par le symbole .

La feuille de route

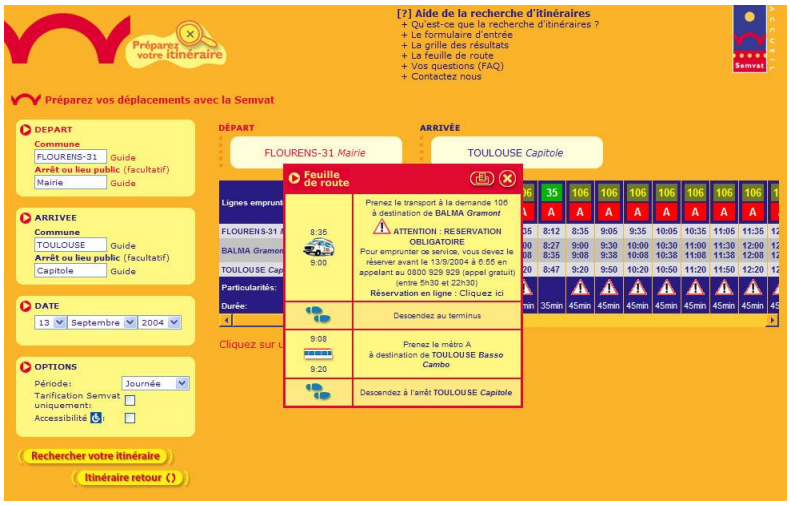


Figure 25: Exemple de feuille de route générée par Synthèse

- Présentation

La feuille de route permet d'obtenir toutes les informations nécessaires pour effectuer un trajet sélectionné sur la grille des résultats. En outre, pour certaines lignes à réservation obligatoire (transports à la demande etc.), elle permet de réserver sa place directement, en cliquant sur le lien « cliquer ici ».

- Descriptif du trajet

Le trajet y est décrit clairement, en faisant état des modes de transport et des lignes à emprunter, de la destination des véhicules, des points d'arrêt de descente, des portions de trajet à effectuer à pied, etc.

- Alertes

Toutes les informations de type alerte sont indiquées au fil de la description du trajet, pour notifier les informations de nature exceptionnelle, ou pour inviter à réserver une place lorsque cela est nécessaire. Sur certaines lignes, un lien direct vers la page de réservation est proposé en plus de la description des modalités de réservation.

- Services continus

Dans certains cas, les heures sont remplacées par des plages horaires (Ex : 8 :00 à 12 :00). Ceci indique que la solution proposée est valide en continu sur la plage horaire mentionnée. Dans ce cas, les temps de parcours affichés incluent l'attente aux arrêts.

Architecture générale

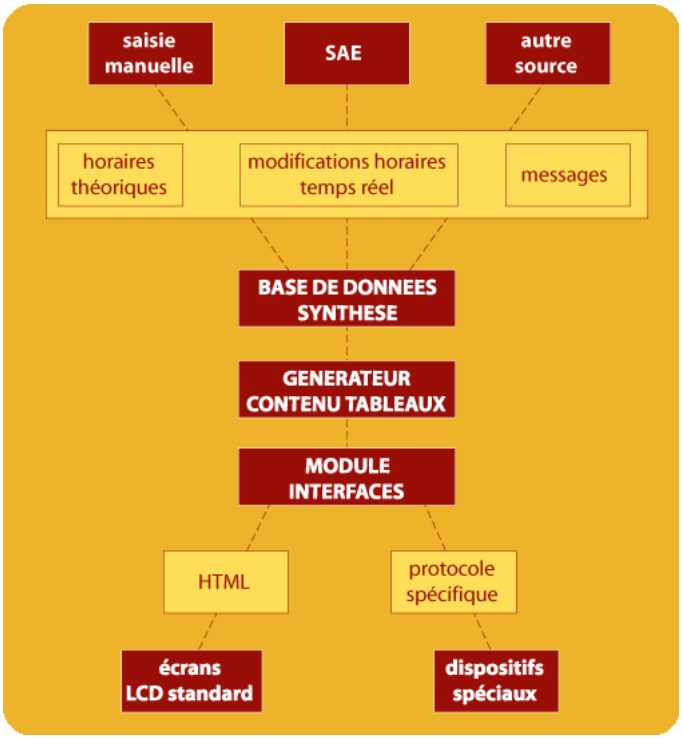


Figure 26: Schéma d'architecture générale de Synthèse

L'architecture générale interne de Synthèse est structurée en couches indépendantes :

1) La base de données (SQLite / PostgresSQL), contenant les tables référentielles, pouvant être remplies par un import Trident issu directement de Chouette (voir le chapitre « données décrivant le réseau TC »), ou d'après d'autres sources

2) La partie « générateur » elle-même subdivisée en deux sous parties :

   a. Un modèle objets « compilé » en mémoire vive, permettant des algorithmes à haute performance

   b. Une bibliothèque d'algorithmes dont le calcul d'itinéraires multimodal

3) Le module « Interfaces » transformant les résultats d'algorithmes en sorties présentables selon un format défini en base de données par un langage fonctionnel type Lisp simplifié, et permettant l'encapsulation récursive de composants

L'architecture d'un système basé autour de Synthèse est composée de deux parties :

1) La partie serveur décrite précédemment

2) Un client permettant d'effectuer des requêtes au serveur via la couche TCP, et retranscrivant les résultats à un flux de sortie (serveur web, matériel dédié, etc.)

Exemples d'architectures :

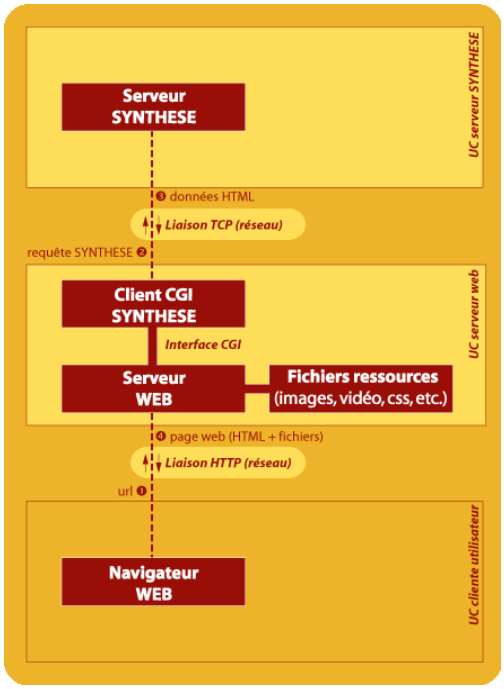


Figure 27: Architecture web via CGI

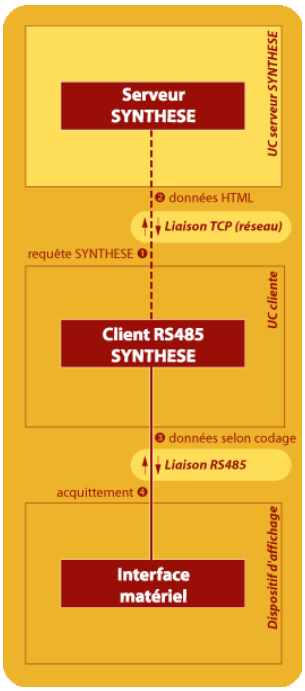


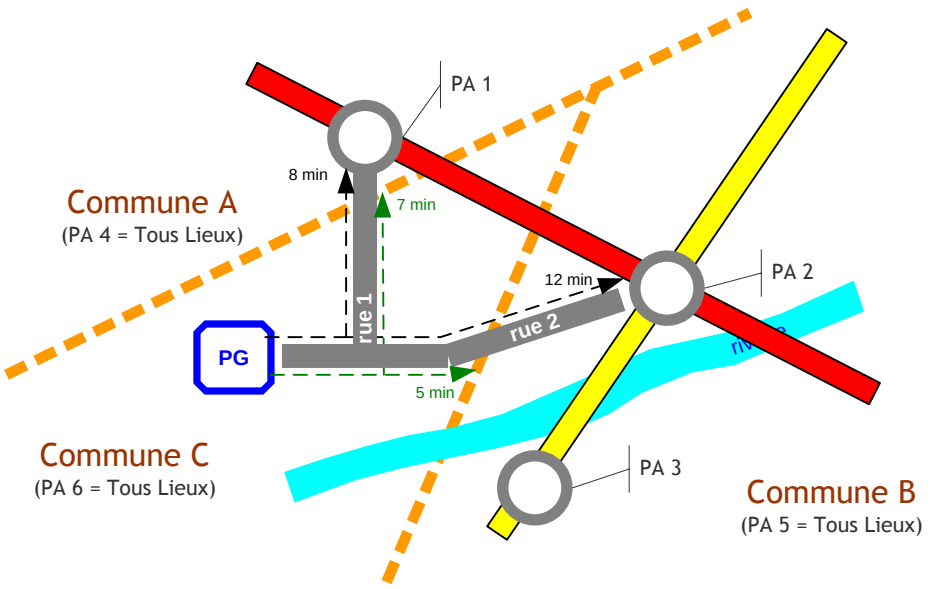
Figure 28: Architecture panneau d'affichage à diodes via client RS 485

Algorithme

L'algorithme de calcul d'itinéraire multimodal effectue les tâches suivantes :

1) Détermination des arrêts atteignables au départ en utilisant la voirie, par parcours intégral du graphe de voirie, dans le respect des limites fournies en paramètres (longueur max à pied, etc.)

2) Détermination des arrêts permettant d'atteindre l'arrivée en utilisant la voirie, par parcours intégral du graphe de voirie, dans le respect des limites fournies en paramètres (longueur max à pied, etc.)



3) Calcul de la première solution depuis les n arrêts de départ, pondérés chacun par leur temps de marche depuis le départ respectif, vers les n arrêts d'arrivée, pondérés chacun par leur temps de marche vers l'arrivée respectif. Une solution est retournée si et seulement si elle permet une arrivée le plus tôt possible, tout en partant le plus tard possible.

4) Calcul de la solution suivante à l'heure de départ ajoutée d'une minute, selon les mêmes critères.

5) Idem jusqu'à trouver une solution dont l'arrivée dépasse de 24h l'heure de l'arrivée de la première solution, ou jusqu'à ne pas trouver de solution

6) Calcul de la représentation en grille (tri des arrêts intermédiaires, avec éventuelle duplication de rangée de tableau si besoin)

7) Transmission des résultats au module d'interface pour affichage

Compléments :

- Il est possible de calculer sur une plage plus étroite que 24 heures
- Il est possible de calculer les solutions de la dernière jusqu'à la première (conseillé pour l'information voyageurs, afin de proposer des répartitions de temps d'attente plus sécurisés)
- Les points de départ et d'arrivée peuvent se trouver sur la voirie ou sur un arrêt. Dans tous les cas le parcours intégral est effectué pour trouver l'ensemble des arrêts atteignables depuis le point spécifié, y compris si le point spécifié est lui-même un arrêt
- Les lignes à réservation obligatoire sont intégrées et ne sont proposées que si leur réservation est effectivement possible (respect du délai de réservation)
- Les services en fréquence sont gérés en natif dans le système
- Les arrêts « tous lieux » représentent des zones géographiques et non des points, afin de modéliser le TAD zonal sans arrêt de prise en charge
- Toute autre description de service peut être ajoutée dans la mesure de la fourniture d'une classe respectant l'interface définie.
- Des contraintes de filtrage sont définies (handicapés, vélos, etc.) : les lignes sont filtrées à partir de ces critères pendant le calcul, et non les solutions a posteriori.

Modèle de données

Le modèle de données d'entrée de SYNTHESE est partiellement représenté sur le schéma suivant. Les tables en rouge représentent les objets de transport public. Les tables en vert les objets de voirie routière, les tables en jaune représentent les contraintes. Les tables en bleu représentent les objets géographiques. Les tables en blanc représentent la liaison entre les points géographiques et les services (route/transport).

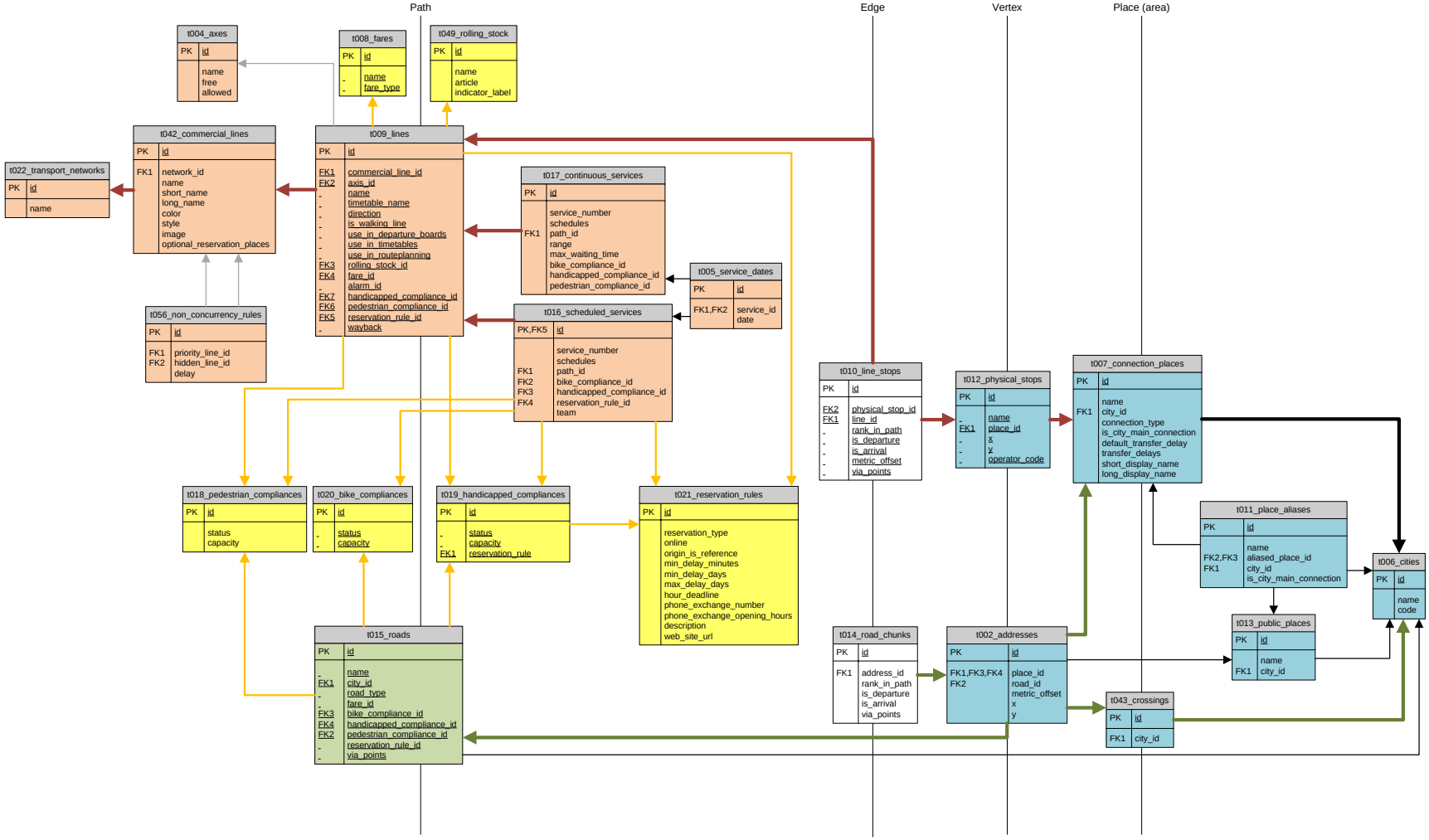


Figure 29: Modèle de données d'entrée de Synthèse

VIII. BILAN ET POURSUITE DES TRAVAUX POTIMART

Ce document de conception technique montre que les composants Open Source utilisés pour **POTIMART** permettent d'ores et déjà de construire des applications complètes et d'effectuer des analyses transport riches.

En termes de modélisation, il s'agit toujours de faire des compromis entre généricité et simplicité de mise en oeuvre : il n'y a pas de standards universels en matière de données transport qui permettrait d'envisager le développement d'un connecteur « plug and play », mais une grande variété de données, s'appuyant certes sur des standards, mais toujours avec des variantes qui nécessitent des adaptations et des procédures de conversion. En matière

- d'offre théorique TC, la normalisation du profil Trident/Chouette qui est en train de se concrétiser en France devrait grandement faciliter la tâche, et est un argument fort pour les outils **POTIMART**. Néanmoins, il faut garder à l'esprit que les normes elles-mêmes évoluent de manière assez rapide, comme en témoignent les travaux en cours concernant Ifopt et Siri au niveau européen. Les choix faits dans **POTIMART** devraient nous permettre de s'adapter à ces évolutions.

- de données de voirie, le modèle POTIMART permet d'utiliser aussi bien des données provenant de base de données de navigations commerciales que de données en libre accès. Ces dernières sont une cible naturelle pour les logiciels Open Source : la qualité des données OpenStreetMap (OSM) s'est améliorée de manière spectaculaire ces derniers mois, comme en témoigne l'ouverture du service OpenRouteService, qui démontre la mise en oeuvre d'un service de calcul d'itinéraire routier s'appuyant sur les données OSM et les spécifications OGC OpenLS.

Au-delà des données de voirie et de transport TC, un travail sur d'autres types de données, et notamment les cheminements piétons, le stationnement, les données de population et de mobilité, serait certainement utile mais ne peut être envisagé qu'à l'occasion d'une étude particulière sur ces données.

En termes d'architecture, l'objectif du projet n'était pas de livrer un progiciel « clé en main », mais des composants interopérables, et d'indiquer la manière dont ils doivent être adaptés pour construire une application en particulier. En termes d'analyse SIG transport, les prochains efforts de développements, qui n'ont pu encore être menés compte tenu des ressources allouées au projet, concernent la mise en place d'une API de calcul d'itinéraire multimodale et des isochrones, a priori en adaptant le logiciel Synthèse au modèle Trident sous PostGIS et au langage de script sont les pistes envisagées. De même, la spécialisation du framework MapFish aux données transport permettra de développer encore plus facilement des applications Web. Par ailleurs, l'application Chouette est elle-même susceptible d'évoluer, et d'être complétée dans les années à venir par d'autres outils, sous l'impulsion du Ministère en vue de soutenir les travaux de normalisation.

En pratique, la prochaine étape consistera pour les partenaires à démontrer et valoriser les acquis du projet, en particulier en termes d'analyse de réseaux, afin de faire émerger de nouveaux projet qui permettront de développer ou améliorer les briques qui manquent encore, pour progressivement construire plus complet. Les partenaires continuent donc activement à échanger avec les utilisateurs potentiels (notamment les services transport des collectivités et les bureaux d'études) et tout commentaire ou question sur le présent document et plus largement sur le projet sont les bienvenus.

IX. ANNEXES

A. Glossaire

AO	Autorité Organisatrice
API	Application Programming Interface
BD	Base de Données
CartoWeb	Libraire Open Source web SIG développée par CampToCamp (remplacée par MapFish)
Chouette	Création d'Horaires avec un OUtil d'Echange de données TC selon le format Trident Européen Format d'échange de données TC normalisées au niveau français, et logiciel Open Source permettant de créer / produire des données à ce format http://chouette.mobi
DSL	Domain Specific Language
IFOPT	Identification of Fixed Objects in Public Transport
IHM	Interface Homme Machine
JourneyPattern	Mission d'un itinéraire
Line	Ligne de transport en commun
MapFish	Libraire Open Source web SIG développée par CampToCamp (Carto Web v4)
OD	Origines/Destinations
OGC	Open Geospatial Consortium
Open Source	La désignation Open Source (source ouverte en français) s'applique aux logiciels dont la licence respecte des critères précisément établis par l'Open Source Initiative, c'est-à-dire la possibilité de libre redistribution, d'accès au code source, et de travaux dérivés.
OSM	OpenStreetMap
PathLink	Tronçon d'une mission
PgRouting	Librairie de calcul d'itinéraires Open Source pour PostGIS
PostGIS	Contraction de PostgreSQL et de GIS, PostGIS est le module spatial qui confère à PostreSQL le statut de SGDBRO spatial. Ce site est le portail français contributif des utilisateurs de PostGIS
POI	Point d'Intérêt
PREDIM	Plate-forme de Recherche et d'Expérimentation pour le Développement de l'Information Multimodale
REST	Representational State Transfer
RI	Recherche d'Itinéraire
Route	Itinéraire d'une ligne
SHP	Fichier de forme ShapeFile
SIRI	Standard Interface for Real time Information
SQL	Structured Query Language
StopArea (Commercial)	Arrêt commercial d'une ligne
StopArea (Physique)	Arrêt physique d'une ligne
StopPoint	Arrêt topologique
SYNTHESE	SYNTHESE (SYstème Numérique de Traitement des Horaires En Situation normale et Exceptionnelle) est une suite logicielle articulée autour d'un serveur de base de données dédié au transport public développé par la Réseaux Conseil Solutions Informatiques
TC	Transports Collectifs
Trident	TRansport Intermodality Data sharing and Exchange NeTwork
VP	Voiture Particulière
VehicleJourney	Course d'une mission (avec horaires)
W*S	WMF, WFS, WCS (protocoles OGC de présentation web de l'info géographique)

B. Références

Expression des besoins des utilisateurs potentiels - Référence : Potimart-Phase2-ExpressionBesoinsCasUtilisation-V4.pdf

Données :

- *NAVSTREETS Reference Manual de NavTeq*
- *MultiNet User Guide de TéléAtlas*
- *GDF : www.ertico.com/en/links/links/gdf_-_geographic_data_files.htm*
- *IFOPT : http://www.naptan.org.uk/ifopt/index.htm*
- *OpenStreetMap : www.openstreetmap.org*
- *SIRI : http://www.kizoom.com/standards/siri*
- *Transmodel : www.transmodel.org/en/cadre1.html*
- *Trident : http://www.predim.org/spip.php?article1087*

*Sites **POTIMART*** : *www.potimart.org*

Site des partenaires POTIMART:

- *www.mobigis.fr*
- *www.camptocamp.com*
- *www.dryade.net*
- *www.reseaux-conseil.com*
- www.cete-mediterranee.fr

Site PREDIM: www.predim.org

*Sites relatifs aux logiciels utilisés dans **POTIMART** (Transport, GOS, graphes, Open Source) :*

- *http://trac.mapfish.org/trac/mapfish*
- *Chouette : http://adullact.net/projects/chouette - www.chouette.mobi*
- *PostGIS : www.postgis.org*
- *PgRouting : http://pgrouting.postlbs.org*
- *www.osgeo.org*

C. Liste des figures

D. Liste des tables