Conception et réalisation d'un système centralisé de données

Benjamin Fracas

le cnam

RATP

CONSERVATOIRE NATIONAL DES ARTS ET METIERS
PARIS

Mémoire présenté en vue d'obtenir
Diplôme d'ingénieur AISL (Architecture et Intégration des Systèmes Logiciels)

Spécialité : INFORMATIQUE

Présenté par

Benjamin FRACAS

Conception et réalisation d'un système centralisé de données

Soutenu le **15/10/2015**

JURY

PRESIDENT :
Mr Nicolas TREVES

MEMBRES :
Mr Marc SCHILDKNECHT
Mr Jean-Marc MINAS
Mr Jean-Marc FARINONE
Mr Nicolas TRAVERS

Tables des matières

Remerciements

Les premières personnes que je remercie sont mes collaborateurs avec qui j'ai pu travailler depuis mon arrivée à la RATP. Ils m'ont beaucoup soutenu et encouragé tout le long de ma formation. Étant l'un des plus jeunes de l'équipe, ils m'ont aussi orienté et conseillé sur les méthodes à employer pour ce mémoire.

Je tiens aussi à remercier la RATP qui m’a permis de réaliser mon projet de fin cycle dans ses locaux.

Je remercie mon tuteur académique Mr Nicolas Trèves qui a su me donner des conseils et des orientations pour ce mémoire. De même pour les cours qu'il m'a enseignés.

De longs remerciements pour les enseignants du Centre National des Arts et Métiers (CNAM) pour leurs pédagogies et leurs expertises.

Je souhaite remercier Mr Nicolas TREVES d'être présent(e) à ma soutenance en tant président du jury ainsi que Mr Marc SCHILDKNECHT, Mr Jean-Marc MINAS, Mr Jean-Marc FARINONE et Mr Nicolas TRAVERS en tant que jury.

Pour finir, je souhaite remercier particulièrement ma femme, ma famille et mes amis pour leurs soutiens durant ces six années de cours et pour la réalisation de ce mémoire.

Glossaire

Abréviation	Description
AFQ	Equipements d'affichage
AJAX	Asynchronous JavaScript and XML
BD	Bases de données
CDV	Circuit de voie
CentOS	Community enterprise Operating System
CMP	Compagnie des chemins de fer métropolitains de paris
CNAM	Conservatoire National des Arts et Métiers
CORBA	Common Object Request Broker Architecture
CSV	Comma-separated values
CT	Conduite de Transport
DCOM	Distributed Component Object Model
ESAE-M	Entretien des Systèmes d'Aide à l'Exploitation du Métro et de la distribution d'énergie
ESMC-M	Entretien de la Signalisation et Modes de Conduite du Métro
ESO	Equipements, Stations et Ouvrages d'art
GDI	Gestionnaire des Infrastructures
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
IHM	Interface Homme Machine
IMS	Ingénierie de Maintenance Systèmes
IP	Internet Protocol
IPL	Ingénierie de maintenance, Projets et Logiciels
JSON	JavaScript Object Notation
MER	Métro express régional
MOA	Maîtrise d'ouvrage
MOE	Maîtrise d'œuvre
MSR	Maintenance des Systèmes du RER
MTBF	Mean Time Between Failure
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
OFT	Offre de transports
ORB	Object request broker
OS	Operating System / Système d'exploitation
PCC	Poste de commande centralisée
RATP	Régie Autonome des Transports Parisiens
RER	Réseau Express Régional
REST	Representational State Transfer
REX	Retour d'expérience
RFC	Requests for comments

RMI	Remote method invocation
RPC	Remote Procedure Call
SAE	Système d'aide à l'exploitation
SFTP	SSH File Transfer Protocol
SGBD	Systèmes de gestion de base de données
SI	Système d'information
SIT	Système d'information et de Télécommunication
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SRU	Solidarité et renouvellement urbain
STCRP	Société des transports en commun de la région parisienne
STIF	Syndicat des transports d'Île-de-France
STP	Syndicat des transports parisiens
TC	Télécommande
TCP	Transmissions Control Protocol
TDE	Transformation et Distribution de l'Energie électrique
TE14	Télétransmission
TK	Télécontrôle
TOP	Topologie
UDDI	Universal Description, Discovery and Integration
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
WWW	World Wide Web
XML	Extensible markup language

Introduction

Depuis sa création, la Régie autonome des transports parisiens (RATP) a pour mission de fournir aux voyageurs un service de qualité.

Pour cela, elle multiplie les demandes d'évolutions et de créations d'outils afin d'optimiser, d'améliorer et de contrôler la performance du service voyageurs.

La RATP conçoit un référentiel principal des données topologiques pour chaque ligne de transport (RER, Métro, Tramway, etc…).

Sur les lignes A et B du RER, les outils permettant le suivi ont été conçus en dupliquant ou en reproduisant ce référentiel selon les besoins. Cela signifie qu'il y a aujourd'hui autant de référentiels que d'applications. Les données topologiques se retrouvent alors dispersées sur l'ensemble de nos systèmes de suivi. Ceci favorise le risque d'erreurs ou d'incohérences lors d'évolutions ou de mises à jour. Il devient obligatoire de modifier tous les référentiels pour que chaque outil fonctionne.

Le but de ce projet est de simplifier la gestion des données sur nos systèmes et d'apporter des améliorations dans notre cœur de métier qui est la maintenance applicative.

Pour cela, je propose une refonte de la gestion du système et la fourniture des données dans notre architecture. Il est prévu de centraliser l'ensemble des données en un point unique et que chaque donnée n'ait qu'une seule source pour s'assurer de leur intégrité afin de faciliter leur utilisation pour les évolutions ou les modifications applicatives.

Cette centralisation permet également d'améliorer l'accès aux données pour une meilleure diffusion dans nos systèmes mais aussi permettre à l'ensemble des services de la RATP d'y accéder.

On retrouve dans le premier chapitre du mémoire une présentation de l'entreprise dans laquelle j'ai réalisé le projet, ainsi que ses différentes évolutions au cours du temps. Dans ce même chapitre, je présente ma position au sein de l'entité de maintenance.

Le deuxième chapitre rassemble les méthodes et les outils existants me permettant de réaliser le projet.

Le troisième chapitre détaille le contexte du projet existant ainsi que les inconvénients et les avantages de cette architecture.

Le quatrième chapitre est consacré à une description de l'architecture cible et de l'ensemble des tâches effectuées pour mener à bien le projet. Il y a des sous chapitres portant sur la réalisation d'un système de gestion des données et d'un système de gestion des applications contenant un module d'archivage permettant la récupération des données de suivi, un gestionnaire de services web permettant de faire l'interface entre les applications et la base de données et une application web permettant l'affichage des données mais aussi de valider l'ensemble du fonctionnement du système mis en place.

Le cinquième chapitre résume le bilan du projet. Il met l'accent sur les points importants abordés dans ce mémoire et y décrit les différentes tâches restantes à réaliser afin de mener à terme le portage du système.

Le dernier chapitre du mémoire est consacré à une synthèse générale sur ma formation et à la conclusion du mémoire.

1 Contexte professionnel

1.1 La RATP et son histoire

Les transports en commun existent depuis le XVII^{ème} siècle. Ils ont fait l'objet de multiples restructurations. La dernière en date fait suite à une loi en 1941, la compagnie des chemins de fer métropolitains de paris (CMP) fusionne avec la Société des Transports en Commun de la Région Parisienne (STCRP). A cette époque, on enregistre un trafic voyageurs de 1,3 milliard en 1943.
Après la seconde guerre mondiale, pour la réhabilitation et le renouvellement de l'ensemble du réseau ferré et routier, la CMP en charge du réseau est écartée par le ministre des transports de l'époque René Mayer. La multitude de changements favorise la création de la RATP en 1949.

Figure 1 - Premier logo de la RATP

Cette création permet de lancer des chantiers d'amélioration des réseaux. En 1953, on voit apparaître les premières rames de métros pneumatiques qui permettent de diminuer l'intensité sonore.

En dix ans de création, la RATP voit son nombre de voyageurs atteindre les 2 milliards.

Pour une meilleure définition des conditions d'exploitation ainsi que des tarifs, l'état met en place une cellule de surveillance qui se nomme le Syndicat des Transports Parisien (STP) maintenant appelé le Syndicat des Transport d'Îles de France (STIF) depuis 1958.

Le suivi de l'exploitation se développe avec la mise en service du premier Poste de Commande Centralisée (PCC) pour la ligne 1 du Métro en juin 1967. Pour les lignes suivantes, les mises en service s'échelonnent jusqu'en 1975. L'ensemble des PCC se situe à Bourdon (75004 Paris).

Après de longs travaux, apparaît en 1969 le premier transport régional appelé Métro Express Régional (MER) qui devient par la suite le Réseau Express Régional (RER). Le RER est voué à prolonger les différentes lignes de métros afin de desservir un maximum de secteurs sur l'ensemble de Paris.

Figure 2 - Poste de commande centralisée de la ligne A

La RATP met en œuvre avec le STIF de véritables nouveautés qui permettent de contrôler et gérer plus facilement les voyageurs. Il y a l'apparition de la carte orange avec abonnement mensuel et hebdomadaire mais aussi les péages automatiques.

Dans les années 90, la RATP continue d'évoluer en remettant en service le tramway et créant la première ligne de Métro automatique.

Une phase importante dans le développement de la RATP est la création de la loi relative à la Solidarité et au Renouvellement Urbain (SRU) en 2000. Elle permet de pouvoir répondre via des filiales, à des offres en France et à l'étranger.

Depuis 2009, suite à une modification de la législation, la RATP s'est vu supprimer l'exclusivité d'exploitation du réseau pour des raisons de séparation comptable. Ce qui signifie que l'exploitation fera l'objet d'une ouverture à la concurrence. Elle est programmée en 2024 pour les réseaux de bus, en 2029 pour les tramways et en 2039 pour les métros et RER.

Même avec ces changements, en 2014, le groupe RATP est le 5^{éme} acteur mondiale du transport public.

Figure 3 - Logo du Groupe RATP

Le groupe est actuellement présidé par Mme Elisabeth Borne et comptabilise plus de 57000 salariés.

En Île de France, le groupe exploite 14 lignes de métros, 2 lignes de RER, 6 lignes de tramways et 350 lignes de bus.
Il est implanté dans de nombreux pays comme par exemple en Angleterre pour la gestion de l'un des principaux réseaux de bus **[1]**, au Brésil pour la construction et l'exploitation de la ligne 4 **[2]** et dans bien d'autres pays comme l'Algérie, les États-Unis ou l'Arabie Saoudite.

1.2 Ma position au sein du groupe

La RATP est organisée en secteurs d'activités (exploitation, ingénierie, maintenance, juridique…).

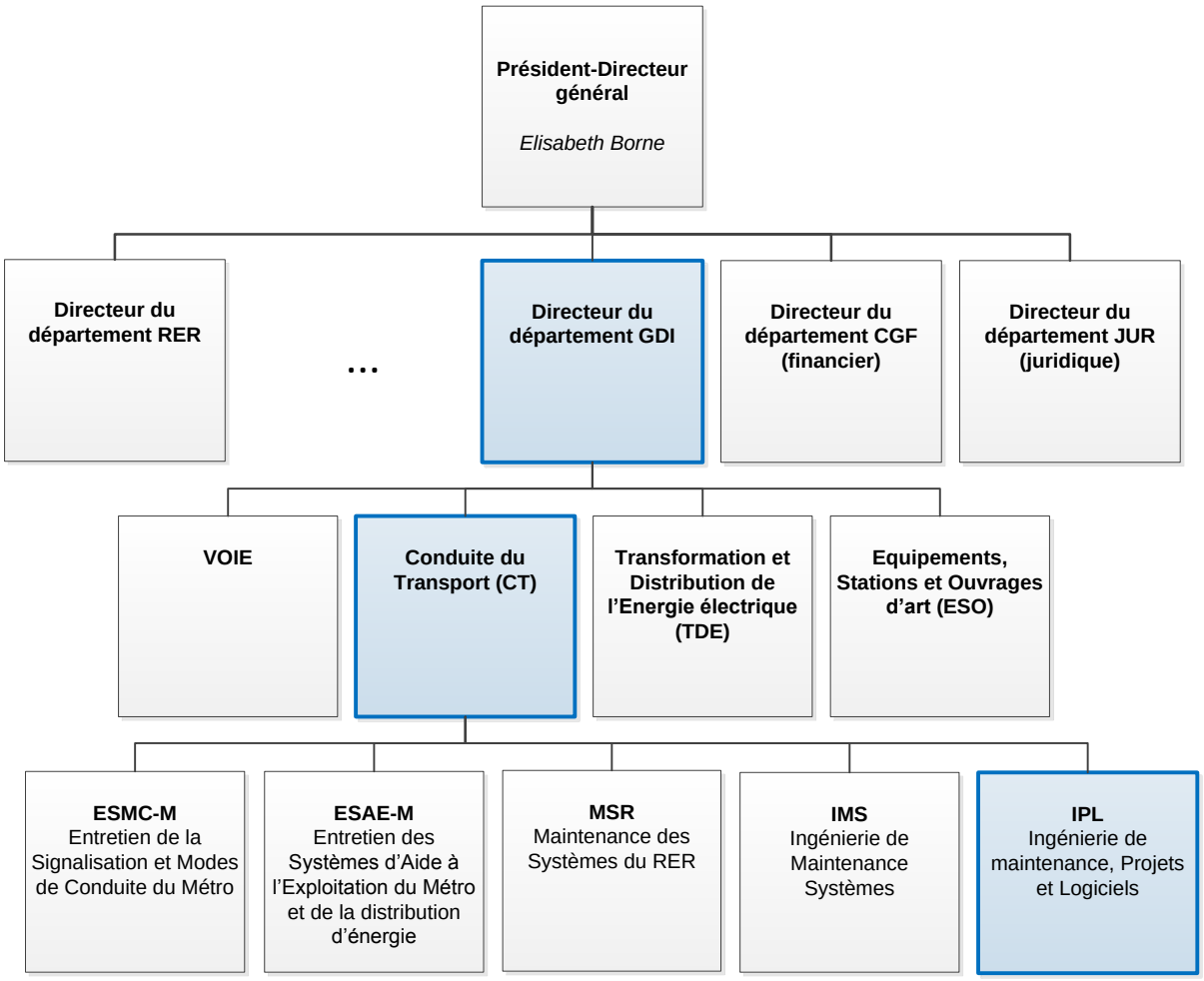


Figure 4 - Organigramme simplifié de la RATP

Je travaille sur des activités du secteur de maintenance informatique qui sont rattachées à un département nommé « Gestionnaire Des Infrastructures » (GDI).

Le GDI a pour mission de gérer l'ensemble des services pour les infrastructures, la maintenance des voies, la maintenance des installations techniques ainsi que l'ensemble des systèmes de contrôle et de suivi sur les réseaux RER, Métro et Tramway.

Au sein de ce département, je suis intégré à l'unité Conduite du Transport (CT).

L'unité CT a pour mission principale la maintenance des systèmes qui produisent et supervisent le transport. Elle intervient dans le domaine de la signalisation ferroviaire, sur les systèmes de pilotage automatique, mais aussi sur la gestion et la supervision du trafic via des applications de suivi des trains et de l'énergie.

Au sein de cette unité, je suis affilié à l'entité Ingénierie de maintenance, Projets et Logiciels (IPL).

L'entité IPL accompagne la maîtrise d'ouvrage (MOA) dans la réalisation des besoins fonctionnels provenant des exploitants et des mainteneurs sur l'ensemble des réseaux ferrés. Elle conçoit, développe et maintient les applications de suivi des trains selon les besoins sur le RER, le Métro et le Tramway.

Cette entité est composée de 12 analystes programmeurs répartis sur l'ensemble des projets, de 4 cadres dont un responsable d'entité, un coordinateur technique et fonctionnel et 2 analystes réseaux. Je suis dans cette équipe analyste programmeur.

Depuis mon entrée à la RATP en 2008, j'ai principalement participé aux développements et aux corrections d'applications de réception, d'affichage et de stockage d' informations venant du suivi des trains du RER.

2 Etat de l'art

2.1 Les projets similaires existants

Dans notre service de maintenance, il n'existe pas actuellement de projet similaire. Certains départements ont mis en place des systèmes centralisés de données pour interpréter et fournir des rapports de production ou de statistiques.

Le premier exemple se base sur la récupération centralisée et la fourniture des données, le département MRB (Matériel Roulant Bus) possède une application appelée GMAO (Gestion de Maintenance Assistée par Ordinateur) qui se trouve être en relation avec des outils pour la gestion des véhicules, des accidents, des machinistes, etc…
Les informations transitent pour lui permettre de générer des rapports journaliers pour chacun des intervenants (machiniste, ressource humaine, manager, etc…)

Le deuxième exemple se base sur la gestion centralisée des données pour la conception et le fonctionnement d'outils. Les nouvelles applications pour le PCC Métro ont été développées par un industriel externe. Ils ont intégré le référentiel transport de la topologie de chacune des lignes du Métro dans une base de données unique. Cette base de données est utilisée pour la configuration et le fonctionnement de chacune de leurs applications PCC Métro. Les données topologiques ne sont toutefois pas accessibles par d'autres applications.

2.2 Les innovations du projet

La véritable innovation de ce projet est de proposer un système de données centralisé permettant une configuration dynamique des outils existants mais aussi un système ouvert permettant de faciliter et simplifier la récupération et la fourniture de données à l'ensemble des applications autorisées.

Ce projet contribue principalement à améliorer notre métier de maintenance dans la partie développement pour nous focaliser sur la partie gestion des systèmes.

2.3 Les solutions utilisées dans ce projet

2.3.1 Les préconisations

La RATP met à disposition des préconisations **[3]** de conception et de développement pour la réalisation des applications. Ces préconisations sont rédigées par le département SIT (Systèmes d'Information et de Télécommunication) qui est le référent dans le domaine informatique. Les préconisations sont valables pour l'ensemble des départements de la RATP. Elles permettent d'uniformiser les architectures au sein de l'entreprise et de privilégier des solutions conformes aux normes et aux standards.

Pour la réalisation de ces applications, la RATP a adopté l'utilisation de l'architecture « N-Tiers ».
Ce type d'architecture est découpé en plusieurs couches :

- Affichage/Présentation : partie de l'application visible par l'utilisateur (IHM)
- Applicative/Coordination : partie fonctionnelle (gestion des services, gestion des erreurs applicatives et techniques, gestion de la traçabilité et des accès).
- Données

Par la suite, les préconisations proposent en fonction des besoins l'utilisation de 2 types de modèle. Il y a le client léger (RIA : Rich Internet Application) et le client riche (RDA : Rich Desktop Application).

Le client léger est accessible uniquement par un navigateur web (internet explorer) et le client riche est installé sur le poste de travail.

Les préconisations portent aussi sur les technologies de développement (outils, méthodes et langages). Dans mon cas, j'utilise principalement des clients légers, des services web et une base de données.

La préconisation adopte pour les serveurs web « Apache » (langage PHP) et « Apache Tomcat » (langage J2EE) avec l'utilisation du protocole http pour les clients légers. Elle utilise REST pour les échanges d'informations entre applications hétérogènes et elle adopte 2 SGBD homologués qui sont par ordre de préférence PostgreSQL et Oracle (sous licence).

2.3.2 Les méthodes et les technologies utilisées

Pour la mise en œuvre de ce projet, j'ai utilisé des outils venant principalement du libre. Ayant des directives pour uniformiser nos outils pour se rapprocher des préconisations établies à la RATP, le système d'exploitation (OS) utilisé est CentOS 6.5 et le SGBD est PostgreSQL.

Une messagerie propre à la RATP qui se base sur le protocole de communication TCP/IP est utilisée pour la communication entre les clients (archivage) et le serveur de suivi. (*cf chapitre **3.3.2**)*

Pour la gestion des données (insertion, modification, suppression et récupération) dans la base, j'utilise des services web. L'architecture REST est mise en place pour les réalisés. (*cf chapitre **2.3.3.3.4***)

Pour la synchronisation des données dans la base, j'utilise un système de réplication synchrone Maître-Esclave (*cf chapitre **2.3.3.2.2***) par l'intermédiaire du middleware Pgpool II (*cf chapitre **4.2.2.5***).

Le reste des outils ont été choisis pour des besoins de performance et de simplification.
Les technologies utilisées dans ce projet :

- le langage BASH/PERL/SQL pour l'insertion des données topologiques et de l'offre de transport dans la base de données.
- le langage PERL pour la récupération des données venant du suivi.
- le langage Javascript/Node.js pour le traitement des services web et l'affichage temps réel de l'information.
- le langage HTML pour l'affichage des informations sur le navigateur web.

2.3.3 L'analyse des solutions possibles

2.3.3.1 Présentation

Pour comprendre les choix techniques retenus, un état de l'art est fait sur les solutions où plusieurs solutions sont possibles pour la réalisation de ce système. L'étude se base sur la réplication de données avec le SGBD PostgreSQL **[4, 5]** et la réalisation des services web.

2.3.3.2 La réplication des données

2.3.3.2.1 Réplication multi-maîtres

Des solutions multi-maîtres existent mais nécessitent un travail rigoureux dans la gestion des requêtes. Cette méthode améliore la disponibilité mais augmente le risque d'incohérence lors de l'insertion des données.
Un nœud correspond à un accès dans la base. Dans un système multi-maître, il y a autant de nœuds qu'il y a de serveurs. Ils ont chacun la possibilité d'interagir sur les bases.

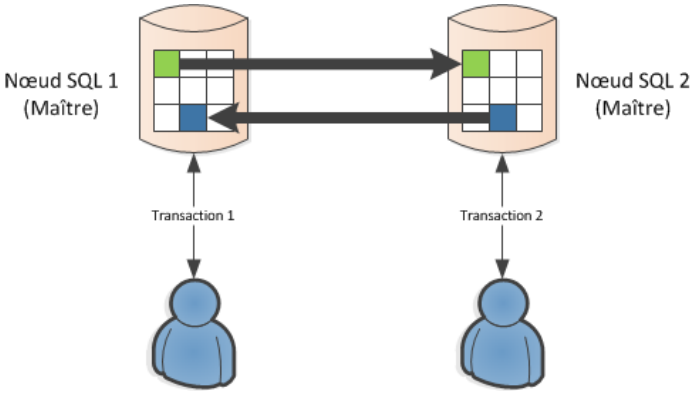


Figure 5 - Réplication multi-maître

Ce sont des systèmes utilisés pour un besoin d'accès en lecture important. Cette solution n'est pas retenue.

2.3.3.2.2 Réplication maître-esclave synchrone

Pour une architecture maître-esclave, le principe d'insertion des requêtes transitent par le serveur maître qui se charge de mettre à jour le ou les serveur(s) esclave(s).
La méthode synchrone permet la mise à jour des bases de données instantanément. La transaction des données n'est validée qu'après acquittement de la transaction du maître et de l'esclave au serveur (maître) ayant exécuté la requête.

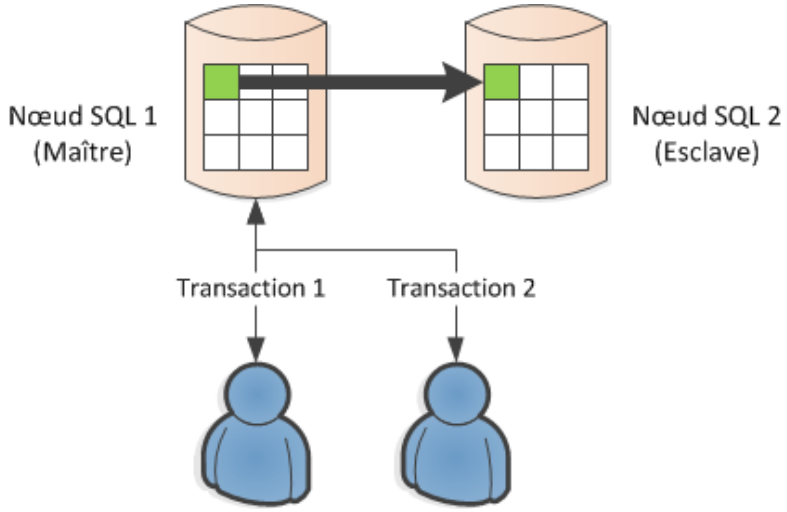


Figure 6 - Réplication maître/esclave synchrone

2.3.3.2.3 Réplication maître-esclave asynchrone

La méthode asynchrone permet la mise à jour de bases de données maître et esclave par une méthode de synchronisation intermédiaire. En cas de coupure du système de synchronisation ou de l'un des serveurs, l'ensemble des autres serveurs peuvent continuer à fonctionner. Cependant en cas de relance, il est nécessaire de mettre en place un processus de synchronisation et de gestion de conflit dans le cas où l'un des clients serait amené à être en lecture sur l'un des serveurs esclaves.

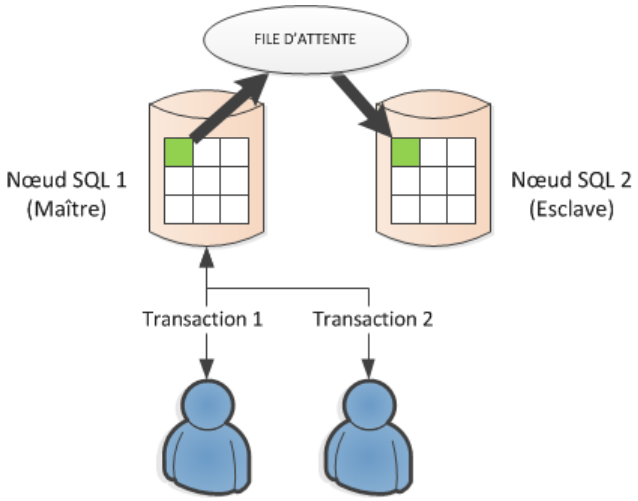


Figure 7 - Réplication maître/esclave asynchrone

2.3.3.3 Les services web

2.3.3.3.1 Historique

Depuis une trentaine d'année, les entreprises essayent de résoudre des problèmes récurrents que sont la distribution de l'information mais aussi l'interopérabilité des systèmes d'information (SI) **[6]**.

La finalité est de permettre aux différents acteurs que ce soit le client, le fournisseur ou toute autre personne d'avoir accès à l'information via un accès web.

Il y a deux méthodes sur le partage d'information.

La première correspond à une architecture développée en interne et connectée à un nombre limité et connu de clients. L'architecture actuelle utilise cette méthode. Cependant, il devient difficile maintenant de fournir correctement les informations lorsque le nombre de clients commence à s'agrandir, l'architecture n'avait pas été prévu de cette manière.

La seconde correspond à l'utilisation de services web, le nombre de clients peut être potentiellement illimité. L'inconvénient de cette méthode provient de l'échange des données à fournir, il faut rendre le système interopérable.

L'utilisation des services web est intéressante pour de multiples raisons. Elles sont décrites par le **World Wide Web Consortium (**W3C[1]) qui normalise leur utilisation pour que les technologies employées soient compatibles entre elles.

La première correspond à produire une interface interprétable par des applications pour permettre aux clients d'accéder aux services et à l'information de manière automatique.

La deuxième est l'utilisation de langage et de protocole indépendant de tout langage de programmation et de toute plateforme d'exécution. La dernière correspond à l'utilisation des standards du web tels que le « HTML » et « XML » qui sont normalisés.

[1] **W3C** : Organisation de normalisation des langages web comme le HTML ou le XML. Créer par Tim Berners-Lee en 1994.

Il existe beaucoup de technologies pour mettre en place des services web. La plupart se base sur des protocoles qui existent depuis une quarantaine d'années.

Les premières technologies de services web se basent sur le protocole de communication Remote Procedure Call (RPC). C'est un protocole réseau qui permet de faire des appels de procédures distantes. Quelques exemples de technologies qui se basent sur RPC : RMI, DCOM et CORBA.

Voici rapidement en détail ce que propose chacune de ces technologies :

- RMI : Remote Method Invocation

Une interface de programmation pour le langage Java qui permet d'appeler des méthodes distantes, sur le principe des Object Request Broker (ORB). C'est un ensemble de fonctions qui envoient et reçoivent des requêtes.

- DCOM : Distributed Component Object Model

Une technique propriétaire de Microsoft qui permet la communication entre des composants logiciels distribués au sein d'un réseau informatique.

- CORBA : Common Object Request Broker Architecture

Architecture logicielle pour le développement de composants et d'ORB.

Le protocole RPC n'étant pas adapté à internet (HTTP), cela pose de réels problèmes de compatibilité et de sécurité entre systèmes. Les proxys et les pare-feux[2] bloquent généralement ce genre de communication. Cette technologie n'est pas adaptée pour un partage simple de l'information, nous ne l'utiliserons pas pour nos développements.

Pour développer un protocole générique permettant à toutes les applications de communiquer sur le web, plusieurs standards basés sur XML ont alors vu le jour.

Il y a le protocole XML-RPC se basant sur RPC. Il apporte une nouveauté sur le fait qu'il utilise les standards du web que sont les HTTP et le XML pour les appels de procédures.

Devenant de moins en moins utilisé du fait que certaines fonctionnalités sont manquantes. Deux méthodes ont été conçues pour améliorer leur utilisation. La première est une évolution de XML-RPC, c'est le protocole « Simple Object Access Protocol » (SOAP). Il a été créé

[2] **Pare-feu** : firewalls

pour apporter de nouvelles fonctionnalités telles que l'ajout d'un mode asynchrone et une gestion avec ou sans état. La seconde modifie la méthode d'orientation des services web en mettant l'accent sur les ressources et non plus sur les objets. Ce n'est plus un protocole mais une architecture. Elle se nomme REST.

2.3.3.3.2 Les standards du web

Les services web utilisent des standards afin de rendre l'échange, la récupération et la lecture de données rapidement et facilement (HTTP, XML, JSON).

Le HTTP (HyperText Transfer Protocol) est un protocole de communication entre un client et un serveur. C'est un protocole normalisé par les normes RFC 7230 à RFC 7237. C'est un protocole qui se base sur le protocole TCP. Les communications s'effectuent sur des supports universels et généralement non filtrés par les pare-feu.

Le XML (eXtensible Markup Language) est un format de données. Il est considéré comme un langage HTML amélioré permettant de définir de nouvelles balises. A l'inverse du HTML, c'est un langage figé par le W3C. Le XML permet de séparer le contenu de la présentation.

Le format JSON (JavaScript Object Notation) permet de décrire des données sous forme de texte. Il permet comme le XML de structurer l'information. C'est un format qui est normalisé par la RFC 7159.

2.3.3.3.3 Le protocole SOAP

SOAP **[7]** est un protocole de type RPC orienté objet basé sur le format XML. Il permet la transmission de messages entre objets distants, c'est à dire qu'il permet à un objet d'invoquer des méthodes objets situées sur un autre serveur.

Il est comme XML-RPC, indépendant de la plate-forme ainsi que du langage de programmation. Il permet l'utilisation de différents protocoles de transport tel que SMTP (Simple Mail Transfer Protocol), contrairement à XML-RPC qui n'utilisait que HTTP. C'est un protocole qui est soutenu par le W3C.

SOAP possède de nombreux inconvénients. C'est un protocole qui devient lent lorsque le nombre d'objets est important. Elle provient du fait que le langage XML impose une structuration des données demandant parfois une très grande quantité de données à échanger. Il doit contenir la définition du contenu du message et la méthode pour le traiter, les différents types de données et l'organisation des requêtes à recevoir.

C'est un protocole qui décrit les méthodes de communication entre applications. On le compare maintenant à un framework plus qu'un protocole générique sur le fait qu'il est très difficile de rendre interopérable plusieurs services basés sur SOAP. Ce qui signifie qu'une modification est nécessaire côté client lorsqu'une modification est réalisée sur l'API principale. Cela rend le système non interopérable et nous oblige à publier la description du service pour son utilisation.

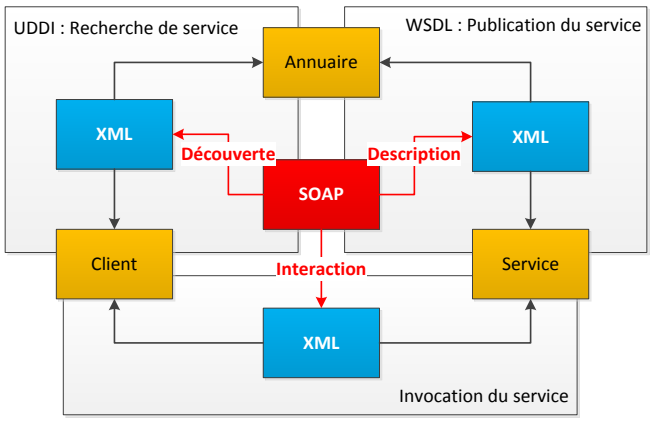


Figure 8 - Le fonctionnement des services web avec SOAP

Pour la mise à disposition de son service web auprès des clients, il existe l'Universal Description, Discovery and Integration (UDDI). On peut le comparer à un annuaire de services. De la même façon, pour l'utilisation d'un service, il est important de décrire son

utilisation. Le langage le plus utilisé est le Web Services Description Language (WSDL). C'est un langage de description qui indique de quelle manière utiliser et interagir avec les services web.

2.3.3.3.4 L'architecture REST

On constate que l'interopérabilité n'est pas simple à mettre en place, il est l'un des points le plus important pour le partage d'information. D'après les travaux de recherche de Roy T. Fielding[3], il choisit de revoir l'orientation côté objet pour une orientation côté ressource.

L'architecture REST **[8]** n'est cependant pas un protocole. Elle adaptée pour le World Wide Web (WWW) et repose naturellement sur le protocole HTTP.

REST utilise les standards du web. Le premier correspond format de données échangé qui peut être le XML ou le JSON. Le second standard correspond à l'Uniform Resource Identifier (URI) pour accéder aux ressources/données. L'URI est l'identificateur d'une ressource. Sa définition doit être réalisée suivant une hiérarchie bien précise, il faut les écrire de l'élément général jusqu'à l'élément unique.

Voici un exemple d'URI pour récupérer des données de topologies.
Dans le premier exemple, je souhaite récupérer l'ensemble des données de la topologie. Pour cela j'utilise la fonction GET avec comme URI :
http://api.servicesweb/topologies

Dans le second exemple, je souhaite avoir l'ensemble des données GARES. Pour cela j'utilise la fonction GET avec comme URI :
http://api.servicesweb/topologies/gares

Pour finir, dans le troisième exemple, je souhaite récupérer seulement une gare avec l'identifiant 45. Pour cela j'utilise la fonction GET avec comme URI :
http://api.servicesweb/topologies/gares/45

Le dernier avantage vient de sa mise en place, l'architecture REST s'installe facilement sur un serveur web, lui permettant d'être rapidement disponible sur le réseau.

[3] **Roy T. Fielding** : Informaticien américain né en 1965 à Laguna Beach dans l'État de Californie aux USA.

Pour chaque URI définit il est possible par l'intermédiaire d'effectuer quatre types d'opération supportées par le protocole HTTP.

- « GET » pour la lecture
- « POST » pour l'écriture
- « PUT » pour la modification
- « DELETE » pour la suppression

3 Contexte du projet

3.1 Présentation

Depuis plus d'une vingtaine d'années, le service de maintenance informatique développe et maintient de nombreuses applications qui s'articulent autour de l'aide à l'exploitation des trains.

Les trois fonctionnalités principales de ses systèmes permettent :

- La récupération et l'échange d'informations sur le réseau entre applications.
- L'agrégation, la transformation et la mise à disposition de différentes sources de données.
- L'affichage des données par l'intermédiaire d'une interface homme machine (IHM).

Chaque application traite en général une des trois grandes fonctionnalités présentées ci-dessus. Pour leurs fonctionnements, elles disposent de données spécifiques aux différentes lignes que ce soit du RER, du Métro et du Tramway.

Les données dont elles ont besoin sont classées par catégories :

- Les informations topologiques de la ligne permettant de retrouver tous les types d'équipements d'une ligne pour les associer entre eux.
- L'offre de transport composée d'un calendrier annuel et des graphiques horaires permettant une visualisation théorique des journées d'exploitation. Le graphique horaire permet de définir par exemple la destination ou les temps de passage aux différentes gares pour chacun des trains.
- Les informations de suivi des trains en temps réel récupérées des serveurs de suivi des trains.

Malgré un découpage, un problème se pose. Ces données ont été principalement stockées en interne pour chacune de ces applications sous forme de fichier, par exemple. Cela signifie qu'une même donnée est dupliquée autant de fois qu'il y a d'applications. Il y a un risque qu'une application n'ait pas les données à jour en cas de modification.

Ce mémoire a pour but de présenter un projet de mise en place d'une nouvelle architecture avec une base de données unique permettant de définir un point d'entrée central pour

l’accès aux données. Ce projet ne s’occupe que de l’architecture de la ligne A du RER, en sachant que l’architecture de la ligne B est exactement fait des mêmes systèmes.

Ce projet remet en cause en grande partie l’ensemble des applications ayant un besoin de données. Cependant, un travail en amont a été nécessaire afin d’élaborer une solution simple pour la mise en place du système sans interférence avec l’existant.

3.2 Organisation du projet

Avant ce projet, je réalisais la maintenance et les évolutions de notre système de données existant. Je rencontrais beaucoup de difficultés à répondre aux nouvelles demandes d’évolutions pour la récupération d'informations de la part de plusieurs services. Avec l'afflux constant des demandes, il m'était impossible d'envisager une refonte dans le même temps.
J’ai profité de mon projet de mémoire pour proposer à mes responsables une refonte complète du système de données ainsi qu’une amélioration pour accéder à ces données.

L’entité IPL disposant d’une équipe de maintenance logicielle peu nombreuse, il n’y avait pas la possibilité de faire intervenir plusieurs personnes sur le projet.
Je l’ai donc réalisé entièrement seul (étude, conception, développement et tests). Une personne est venue en renfort pour reprendre les demandes d'évolution sur le système existant.
Une réorganisation de mes activités professionnelles a été effectuée pour me permettre d’exécuter le projet dans les meilleures conditions. J’ai dû cependant en conserver une partie, ce qui a interrompait régulièrement les phases du projet.

L'ensemble de l'architecture existante est composé d'un ensemble d'applications.
Il est obligatoire dans cette refonte que l'ensemble des fonctionnalités existantes restent en place.
Il m'était impossible de finaliser l'ensemble de toutes les fonctionnalités, j'ai donc choisi de réalisé non pas une fonctionnalité complète mais un ensemble de fonctionnalités permettant de réaliser une chaîne complète du suivi des trains, de la récupération à l'affichage.

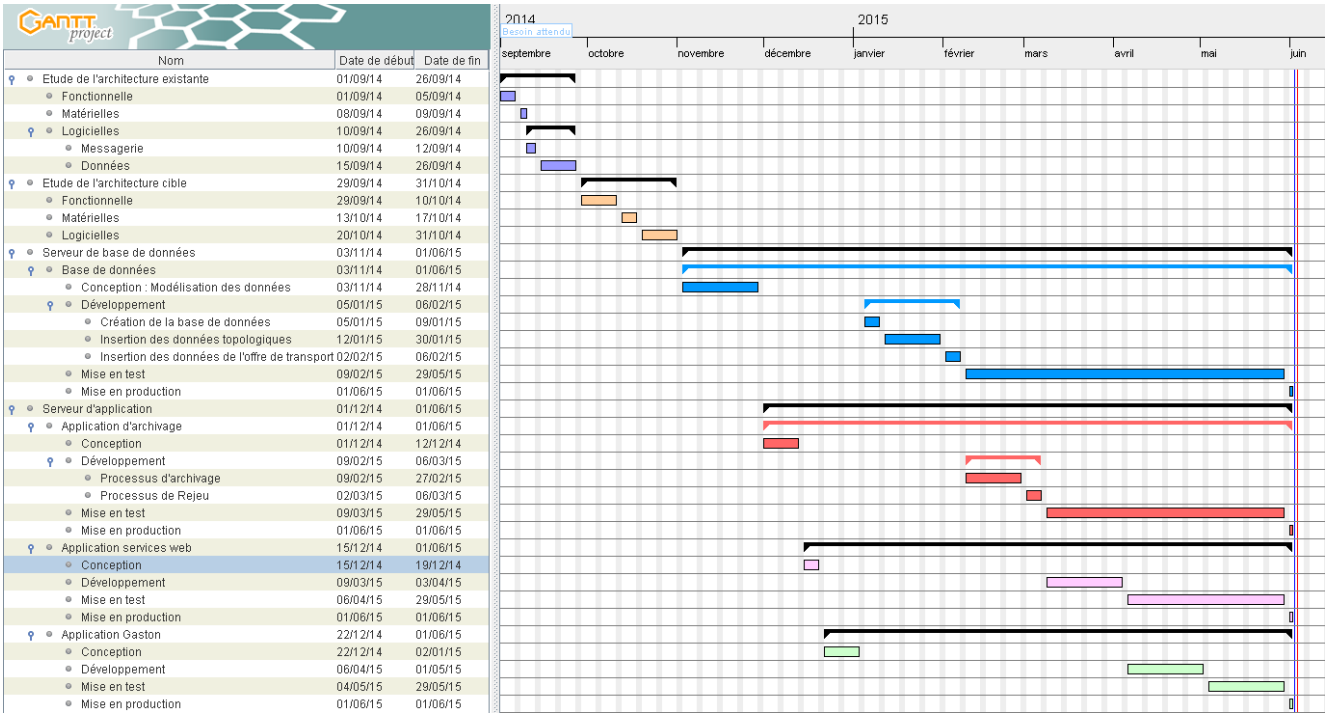


Figure 9 - Diagramme de Gantt du projet

Pour la réalisation de ce projet, je me suis fixé pour date de fin du projet le 1er Juin 2015.

Elle a été organisée par étapes.

Dans ce diagramme est présenté uniquement les délais pour le projet du mémoire. Les fonctionnalités restantes seront traitées ultérieurement.

Le diagramme est composé de six parties. Deux parties pour l'étude de l'existant et de la cible et les quatre parties pour la réalisation des applications.

3.3 Analyse de l'existant

3.3.1 Architecture fonctionnelle

L'architecture liée à la base de données est composée d'un ensemble de systèmes. Ils sont découpés en fonctionnalités permettant de traiter par étapes les données venant du terrain.

Lors de la conception de cette architecture, elle a été pensée pour traiter l'information en trois niveaux.

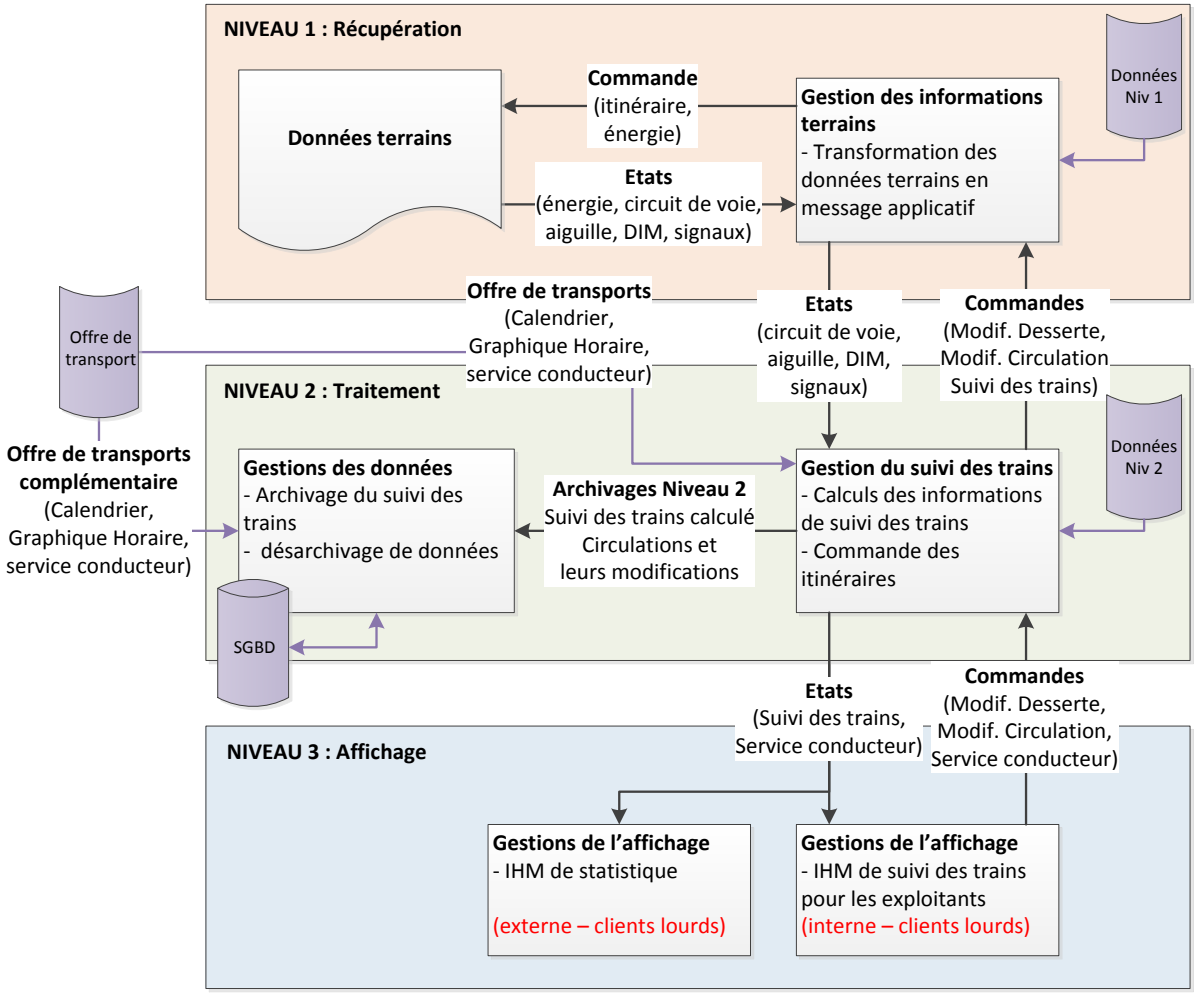


Figure 10 - Architecture fonctionnelle existante

Le premier niveau permet la récupération en temps réel des informations venant du terrain. Les informations/données du terrain représentent les états de chaque équipement de la ligne (circuit de voie, aiguille, signaux, …Etc).

Elles sont remontées par des armoires de télétransmission appelées TE14 pour être transférées vers une fonctionnalité de récupération Cette fonctionnalité utilise des données

topologiques spécifiques (données de la ligne) sous forme de fichiers pour différencier les types d'états qu'il reçoit. Elle renvoie ensuite l'état à la fonction de suivi des trains.

Le second niveau permet le traitement de l'information.
Les états sont réceptionnés par des systèmes de traitements spécifiques tels que le suivi des trains et le traitement de l'énergie. Ils ont pour but de reprendre les informations reçues pour les retravailler mais aussi les compléter. Le suivi des trains utilise aussi des données topologiques qui lui sont propres pour traduire les états qu'il reçoit.
Le suivi des trains vérifie après traduction des données de leur cohérence avec l'offre de transport qu'il récupère sous forme de fichier. Ces informations sont ensuite envoyées à des systèmes d'affichages. Elles sont par la même occasion mises en base de données, à des fins de stockage mais aussi de désarchivage.

Le troisième niveau a pour but d'afficher des informations par l'intermédiaire d'IHM pour améliorer l'aide à l'exploitation des trains.

Certaines IHM permettent d'interagir sur ces informations :

- L'affichage pour la gestion du suivi des trains (pour les aiguilleurs et les régulateurs). Un dialogue entre applications est nécessaire pour la transmission de commandes de suivi telle que la modification de circulation. L'application se présente sous la forme d'un client lourd.
- L'affichage pour la gestion de l'énergie traction (pour les régulateurs). Un dialogue entre applications est nécessaire pour par exemple la coupure de courant sur une portion de voie. (non représentée sur le schéma)

D'autres IHM permettent un suivi d'analyse d'informations uniquement (sans interaction). Par exemple, il existe une application sous forme de client lourd qui compare l'offre de transport théorique et l'offre de transport en temps réel.
Le projet présenté dans ce mémoire participe à l'évolution et au portage de cette application.

3.3.2 Architecture logicielle

L'architecture est composée de plusieurs applications communiquant entres elles pour le bon déroulement du suivi des trains.

Le schéma détaille uniquement les fonctionnalités de niveau 2 et 3, la partie niveau 1 n'est pas représentée. Le niveau 1 n'est pas impacté par le projet.

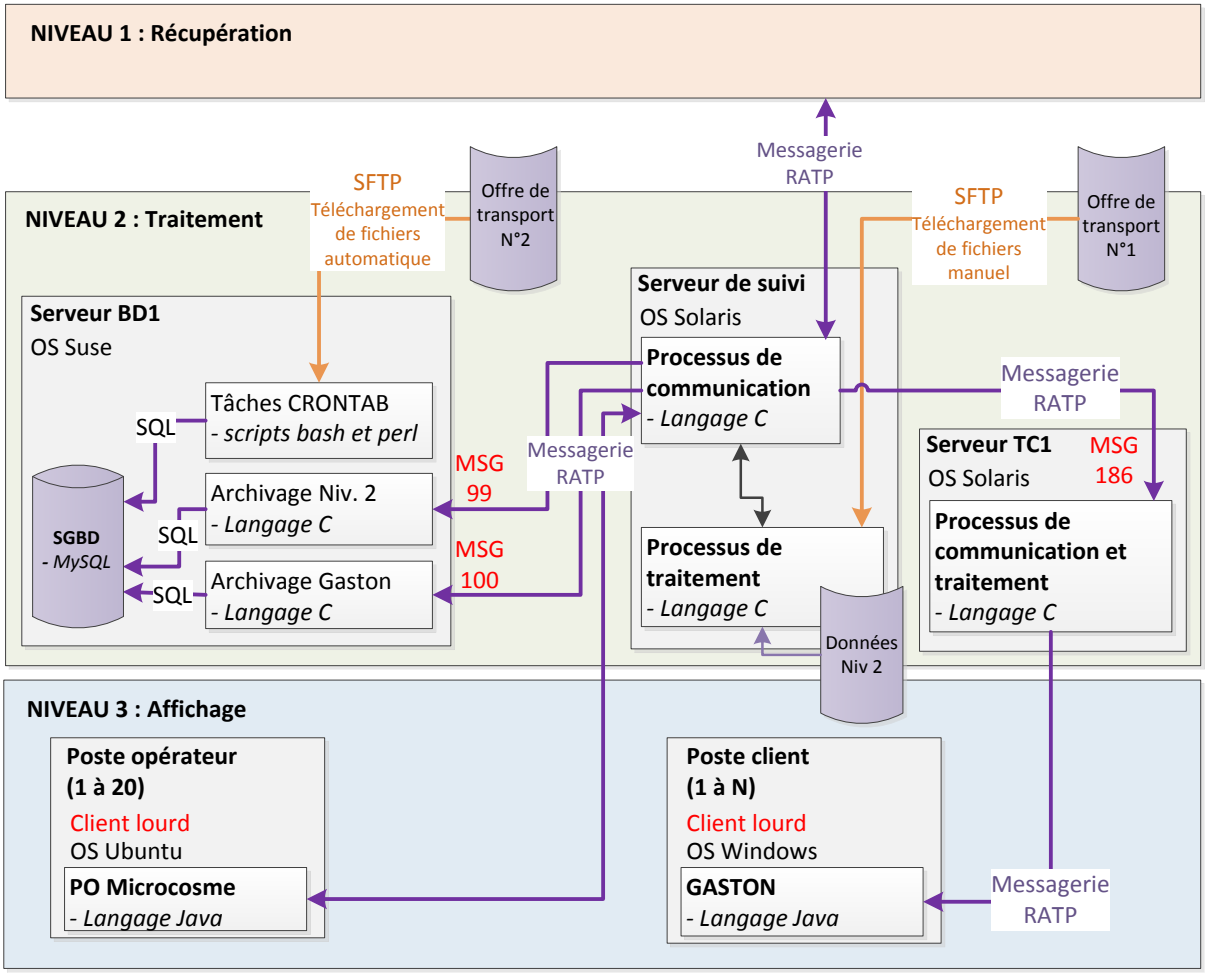


Figure 11 - Architecture logicielle existante

Après récupération, le niveau 1 transforme les états sous forme de messages applicatifs pour les envoyer au serveur de suivi. Les messages utilisent sur le protocole TCP.

Les serveurs de suivi fonctionnent avec le système d'exploitation (OS) Unix Solaris 2.8. Ils ont pour fonction principale la gestion du suivi des trains. Cela consiste à réceptionner les informations du terrain (niveau 1) par l'intermédiaire du processus de communication et les compléter par les données de l'offre de transport et des données topologiques via le processus de traitement.

L’offre de transport est récupérée manuellement par SFTP pour être mise à disposition sur le serveur de suivi pour le processus de traitement. Elle est aussi insérée dans le serveur de base de données par des scripts exécutés automatiquement chaque matin.

Après traitement, le serveur de suivi envoie ses données agrégées à tous les postes clients utilisés par les exploitants pour effectuer le suivi des trains par une interface graphique (clients lourds en interne du PCC), à la base de données pour stocker les informations et au serveur TC1 qui réalise l’interface avec tous les clients lourds en externe du PCC. J’intègre le serveur TC1 à la fonctionnalité de gestion de suivi du fait qu’il ne fait que transiter les messages et ne réalise pas de stockage.

L’échange des données dans toute l’architecture fonctionne par une messagerie mise en place en interne RATP utilisant le protocole TCP. J’ai représenté sur ce schéma les principaux messages de suivi des trains important dans ce projet : MSG099, MSG100 et MSG186. (*cf MSG099 : Le message de suivi et de gestion d'exploitation et MSG100 : Le message de suivi des trains*). Il existe une centaine de messages permettant la gestion complète de l’exploitation des trains.

Cette messagerie est utilisée par l’ensemble des applications développées dans notre service de maintenance.
Le protocole TCP/IP **[9]** fonctionne par segment de 32bits. Chaque segment est composé d’une entête composée de plusieurs attributs pour établir une connexion (en blanc) et d’une partie « données » (en jaune).

<table>
<tr><th>Bit 0</th><th>4</th><th>10</th><th>16</th><th>31</th><td rowspan="8">ENTETE</td></tr>
<tr><td colspan="3">Port Source</td><td colspan="2">Port destination</td></tr>
<tr><td colspan="5">Numéro de séquence</td></tr>
<tr><td colspan="5">Numéro d'acquittement</td></tr>
<tr><td>Taille de l'en-tête</td><td>Réservé</td><td>Codes</td><td colspan="2">Fenêtre</td></tr>
<tr><td colspan="3">Somme de contrôle</td><td colspan="2">Pointeur de données urgentes</td></tr>
<tr><td colspan="5">Options éventuelles</td></tr>
<tr><td colspan="5">Données</td></tr>
</table>

Figure 12 - Détail d'un segment TCP

Pour l’échange de données entre applications internes RATP, nous utilisons le champ « données ». Il va nous permettre d’envoyer des messages de plusieurs types. Pour cela,

nous découpons le champ « données » en deux sections, il y a une partie générale appelée « entête » et un contenu appelé « partie utile ».

L'entête permet de définir le type de message échangé et la partie « utile » de typer les données liées à un événement.

Champ	Taille (octet)	Valeur	Type
Magic Number	4	(A5 A5 A5 A5) hexa	sans
Numéro du message	2		Entier
Configuration	2	0 : **rien**, 1 : **EXP**, 2 : **TES**, 3 :**DEV** , 5 : **RED**	Entier
Nom symbolique de la machine émettrice	4		Caractère
Nom symbolique de la machine destinatrice	4		Caractère
Longueur utile du message	4	1 à n en octets	Entier
Numéro de réseau	2		Entier
Numéro de séquence	2	zéro binaire	Entier
longueur de la séquence	4	zéro binaire	Entier
Heure d'émission	6	hhmmss (caractères ASCII)	Caractère
Réservé	2	Non utilisé (zéro binaire ou espace)	sans

Tableau 1 - Messagerie interne : détail de l'entête des messages

En cas de segment TCP moins grand que notre message, le « magic number » nous permet de récupérer le restant dans le second segment et de nous repositionner sur le second message.

```
BD1:~ # tcpdump -i eth0 -xX -s 0 port 2672
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
12:59:56.453933 IP SE2.32802 > BD1.nhserver: P 1284059901:1284059989(88) ack 332624218 win 24820
        0x0000:  4500 0080 3b49 4000 4006 eee1 c009 c802  E...;I@.@.......
        0x0010:  c009 c837 8022 0a70 4c89 32fd 13d3 715a  ...7.".pL.2...qZ
        0x0020:  5018 60f4 cce1 0000 a5a5 a5a5 0064 0002  P.`..........d..
        0x0030:  5345 3200 4245 3100 0000 0034 0002 0000  SE2.BE1....4....
        0x0040:  0000 0000 3132 3539 3536 0000 3239 3035  ....125956..2905
        0x0050:  3230 3135 3132 3539 3536 0000 4c50 4152  2015125956..LPAR
        0x0060:  5a31 3434 3520 2020 2020 2020 2020 3f30  Z1445.........?0
        0x0070:  3030 3749 3030 3437 3031 3232 3031 3133  007I0047012201l3
12:59:56.453943 IP BD1.nhserver > SE2.32802: . ack 88 win 6432
        0x0000:  4500 0028 5c93 4000 4006 cdef c009 c837  E..(\.@.@......7
        0x0010:  c009 c802 0a70 8022 13d3 715a 4c89 3355  .....p."..qZL.3U
        0x0020:  5010 1920 f6c8 0000                      P.......
```

Figure 13 - Analyse d'une trame TCP d'un échange entre le serveur de suivi et la base de données existante

On retrouve dans cette trame :

- Une partie rouge qui correspond à l'entête TCP.
- Une partie jaune avec le Magic Number.

- Une partie bleue avec les données échangées du message.

Cette lecture est possible suite à l'exécution de la commande linux « TCPDUMP ».

Le serveur de données BD1 utilise l'OS Linux Suse 10.2. Il a pour fonction principale l'archivage des données du suivi. Il utilise le moteur de base de données MySQL.
La réception des données du suivi se fait par des processus d'archivage écrit en C. Il en existe 2 qui sont composés d'un message chacun. Ils utilisent la messagerie interne pour la réception des données du suivi et utilise le langage SQL pour une interaction avec la base de données.

Le serveur TC1 utilise l'OS Linux Suse 10.2. Il récupère les informations de suivi par la messagerie interne et les renvoie au client externe de la même façon.

Les clients d'affichage internes réalisent une interaction (état ou commande) avec le suivi par l'intermédiaire de la messagerie interne. Ils sont aux nombres de 20. Ils utilisent l'OS Linux Ubuntu 12.04.

Les clients d'affichage externes ne récupèrent que l'état du suivi. Ce sont des postes bureautiques utilisant l'OS Windows XP et 7. Ils communiquent uniquement avec le serveur TC1 avec la messagerie interne.

Dans son fonctionnement au quotidien, l'offre de transport journalière peut être récupérer quelques mois en avance, ce qui permet de le mettre à disposition du serveur de suivi et de la base de données.

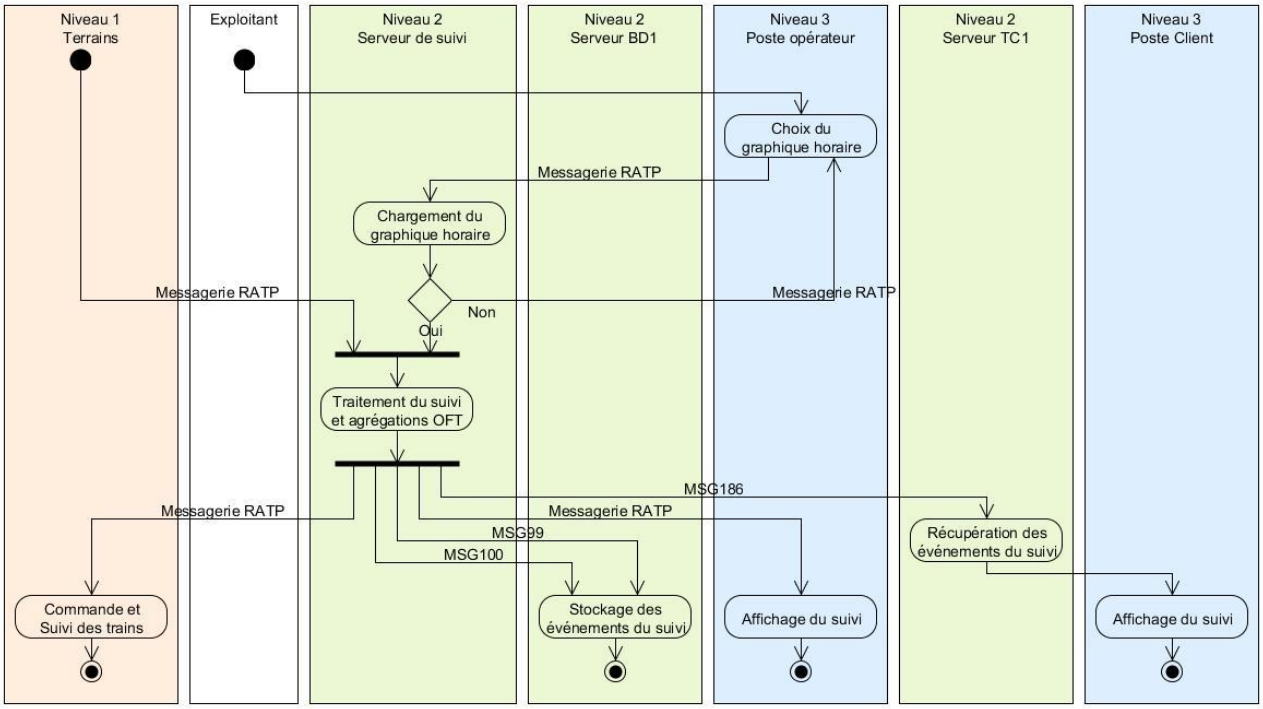


Figure 14 - Diagramme d'activité de l'architecture logicielle existante

Le suivi des trains fonctionne de 4h30 du matin jusqu'à 1h30 le lendemain matin sans interruption. La période d'inactivité permet par exemple la réalisation de travaux sur la ligne/voie mais aussi d'intervenir sur nos systèmes beaucoup moins sollicités qu'en journée.

En début de service, l'exploitant charge un graphique horaire pour la journée de circulation. Il choisit le graphique sur le poste opérateur.

Après chargement, le serveur de suivi reçoit les informations récupérer du terrain (niveau 1). Les informations reçues sont par exemple de type changements d'état (0 ou 1) pour chaque équipement (zone, aiguille, …etc) ou le numéro du matériel d'un train.
Le serveur de suivi complète ses informations avec les données du graphique horaire. Il associe par exemple le matériel avec un nom de mission pour lui définir des propriétés de circulation. Ces propriétés sont les gares et les horaires pour les terminus de départ et d'arrivée, le nom de mission (ex : EKLI22) et bien d'autres.

Après modification, le serveur envoie les informations via la messagerie RATP aux postes opérateurs pour un suivi des trains sous forme graphique. Il les envoie aussi à la base de données. La base de données récupère les messages de suivi (MSG99 et MSG100) par l'intermédiaire de processus d'archivage qui font office d'interface entre le suivi et la base de

données MySQL. Pour finir le serveur de suivi envoie les informations au serveur TC1 pour les regrouper afin de les distribuer aux applications clientes externes.

Les postes opérateurs ont une interaction avec le suivi. Ces postes permettent de charger un nouveau graphique horaire pour la journée à l'initiative de l'exploitant.

3.3.3 Architecture matérielle

L'architecture existante est variée et a évolué avec le temps.

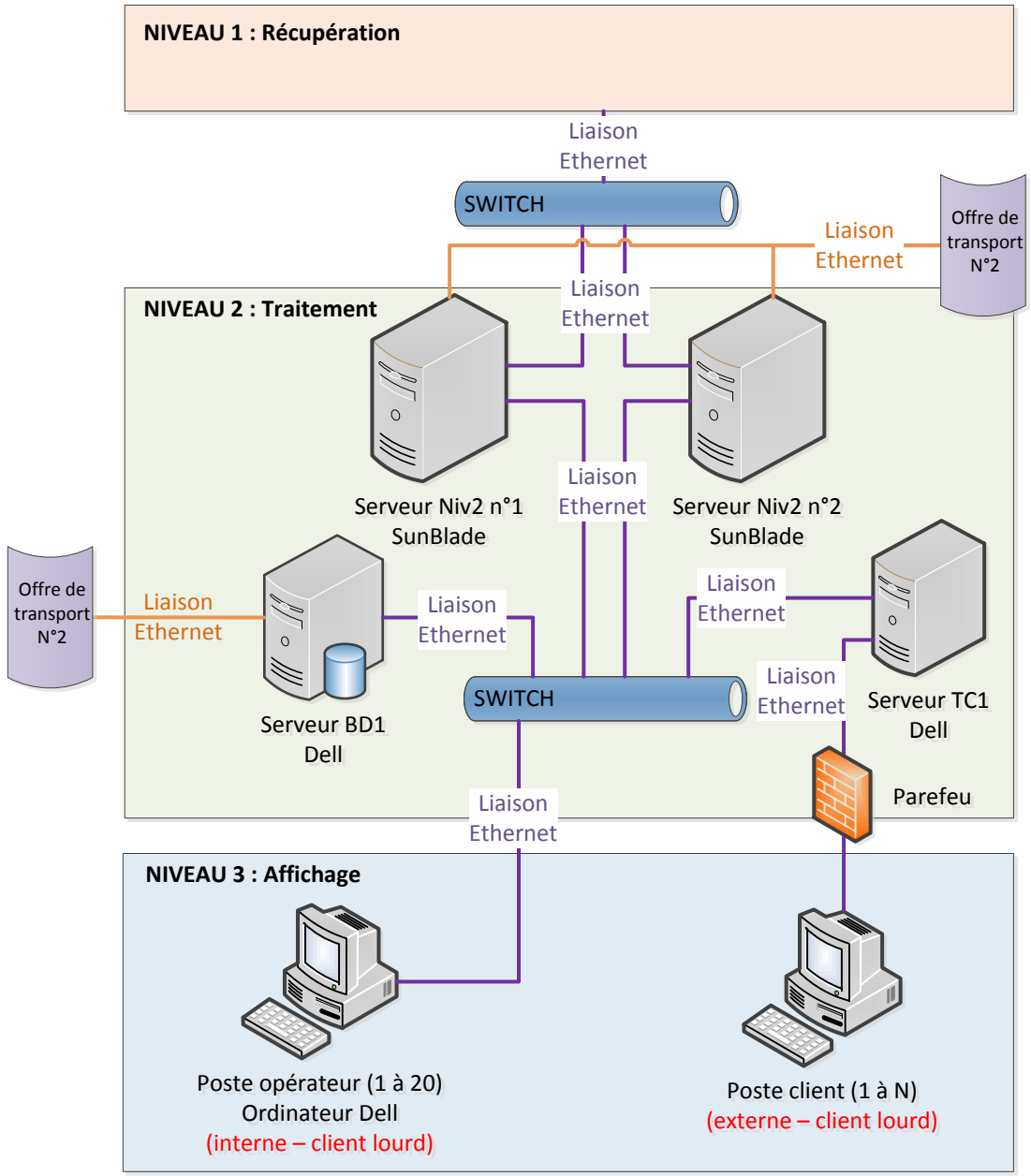


Figure 15 - Architecture matérielle existante

La partie niveau 2 est composée de deux serveurs SunBlade en redondance applicative. Ils permettent le traitement des données terrains pour un renvoi vers l'affichage et le stockage.

Figure 16 – Photo d'un serveur Sun Blade 1000 Workstation

Le stockage est réalisé par un serveur Dell avec un système de disque en RAID5, nommé « BD1 ». La base n'est pas en redondance, la partie RAID5 assure une certaine garantie sur l'archivage des données. Cependant en cas de panne, la base de données ne dispose pas de secours pour prendre le relais.

L'offre de transport est récupérée à deux endroits. La première est historique du fait que le serveur de base de données n'existait pas. La seconde est complétée d'informations supplémentaires non traitées par les serveurs de suivi.

L'architecture a été conçue pour un fonctionnement dans un réseau fermé pour limiter le nombre d'intervenants externes. Pour fournir des informations aux clients extérieurs, les données ont été regroupées sur un serveur nommé « TC1 ». Cela permet de n'autoriser qu'une seule machine à se connecter sur le réseau entreprise (intranet).

L'affichage des données du suivi est découpé en deux parties. Il y a l'affichage du suivi pour les exploitants et pour les clients externes. Pour une interaction avec le suivi, l'exploitant dispose d'ordinateurs Dell en liaison directe avec les serveurs de suivi. Les clients externes correspondent à tous les employés de la RATP qui ont des postes bureautiques en connexion avec le serveur TC1.

L'ensemble des liaisons se font par Ethernet.

3.4 Les raisons du changement

L'avantage de cette architecture est que la récupération, le traitement et l'affichage des données sont séparées. C'est ce qui fait que le système est performant et rapide, puisque chaque fonctionnalité a été optimisée indépendamment.

Malgré de bonnes performances, l'architecture possède certains défauts.

Premièrement, on s'aperçoit que la récupération d'information est de plus en plus longue et complexe à réaliser avec l'ajout permanent de données. Cela est dû principalement à une mauvaise conception de la base. Au départ, c'est un système qui a été mis en place pour palier à des problèmes d'obsolescence d'anciennes bases de données utilisant Informix[4].

Malheureusement pour des questions de temps, la structure de la base a été conçue rapidement. Elle n'offre pas les relations adéquates pour fournir un système efficace pour la mise à disposition des données. Par exemple, il n'existe actuellement aucune relation entre les tables de données, ce qui nous oblige à réaliser des traitements parfois complexe pour relier les informations entre elles.

De plus, il n'existe aucune clé étrangère permettant de définir une certaine intégrité des données entre plusieurs tables.

La refonte complète du modèle de données devient obligatoire pour améliorer l'insertion et la recherche mais aussi ses performances.

Par ailleurs, les différentes applications de l'architecture utilisent leurs propres données topologiques (données Niv 1, données Niv2) mais aussi pour certains une offre de transport spécifique à leur fonctionnement (une offre pour le serveur de suivi et une offre pour la base de données). Les données topologiques et de l'offre de transport se retrouvent éparpillées, favorisant la redondance d'informations mais aussi la possibilité d'avoir des incohérences de données entre applications en cas de maintenance ou d''évolution. Ce qui devient complexe et fastidieux.

Un autre point noir de la base de données vient de sa disponibilité. Il n'existe actuellement qu'une seule base de données, il n'y a donc pas de redondance en cas de panne ce qui engendre des pertes de données non négligeable pour la récupération des données de suivi des trains.

Enfin, les clients lourds présentent de nombreux inconvénients :

- L'un des problèmes majeurs correspond à l'évolution ou la mise à niveau des données contenues en local par chaque client.

[4] **Informix** : système de gestion de base de données relationnelle acheté par IBM en 2001 (**[10]**)

- La mise à jour d'une application n'est effective que lors de son redémarrage. Ceci pose un problème lorsque les personnes la laissent en exécution. Ils peuvent se retrouver en décalage avec la dernière version.
- Il faut prendre en compte la compatibilité de l'application suivant les systèmes d'exploitation.

Pour toutes ces raisons, le système doit être amélioré. Il ne répond plus à la demande initiale qui est d'échanger rapidement et facilement l'information. Cela va améliorer également la maintenance et la disponibilité du système.

Il est également nécessaire de porter l'application sous forme de client léger. Les mises à jour applicatives sont beaucoup plus simples. Il n'y a que le serveur à mettre à jour. On se retrouve avec des clients qui sont toujours à jour par rapport au serveur du fait que c'est ce dernier qui fournit l'interface graphique.
De plus, en appliquant les normes de développement, les applications n'ont aucun problème à être compatibles sur tous les navigateurs.
Cette méthode modifie aussi la démarche d'accès à l'application. Avec les clients légers, il nous est maintenant possible d'augmenter le nombre de clients qui utilisent l'application.

4 Mise en œuvre du projet

4.1 Architecture fonctionnelle de la cible

La cible ne modifie pas le fonctionnel existant. Le traitement de l'information se réalise toujours par niveau. Cependant, certaines méthodes évoluent pour améliorer et simplifier l'architecture existante. Pour rappel, le niveau 1 n'est pas impacté par le projet.

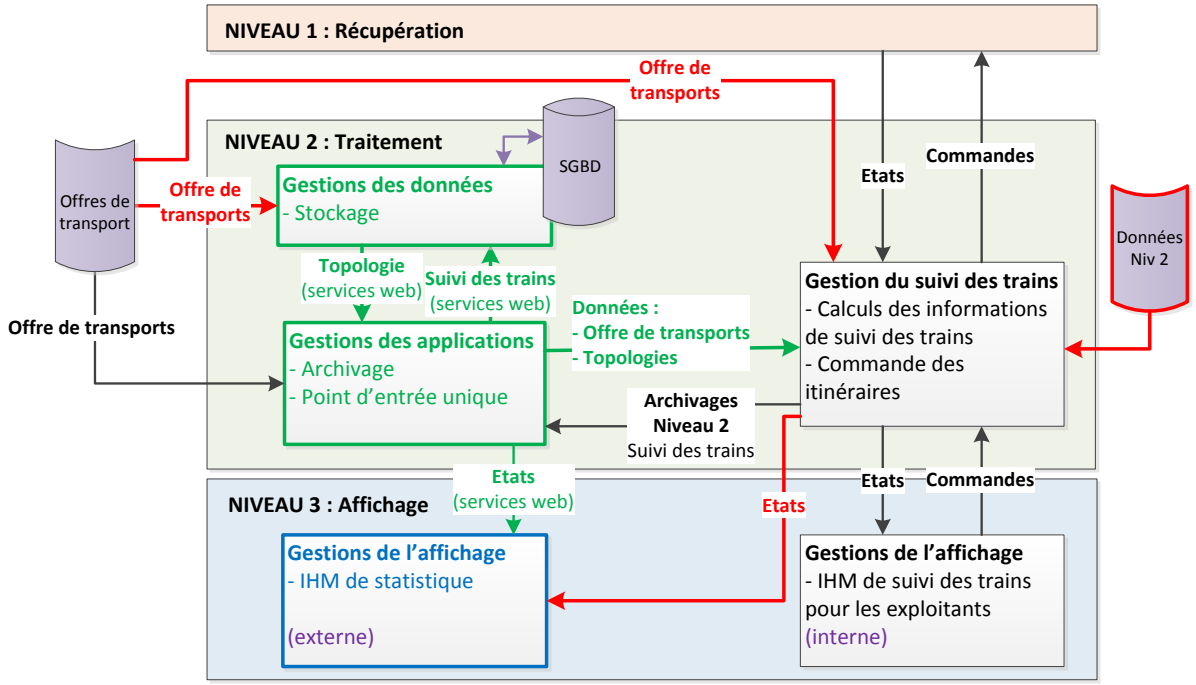


Figure 17 - Architecture fonctionnelle de la cible

Sur ce schéma :

- En vert : les éléments qui sont ajoutés
- En rouge : les éléments qui sont supprimés
- En bleu : les éléments existants qui sont portés sous un autre format

Les principaux changements se font sur la gestion des données. Actuellement l'offre de transport ainsi que les données de topologies sont propres à chaque application. Je vais les centraliser en un point unique dans la gestion des données.

Je mets en place par la même occasion un système simplifié et unique de communication avec la base de données. On va donc créer un système de gestion des applications qui accède aux données par l'intermédiaire de services web. Cette méthode me permet de

mettre à disposition ou de fournir les données à toutes les applications existantes dans l'architecture ayant un besoin de données tel que la gestion du suivi. Elle me permet aussi d'ouvrir l'accès aux données à tout type de client externe (non représenté sur le schéma) que ce soit en lecture principalement ou en écriture.

L'envoi des états du suivi entre la gestion du suivi et l'affichage externe est supprimé du fait qu'il est identique avec les données récupérées par l'archivage niveau 2. Les clients affichage externe récupèrent maintenant les données par l'intermédiaire de la gestion des applications. Cette méthode me permet de supprimer l'envoi multiple de la même information sur différents systèmes.

Pour finir, je réalise le portage d'une application lourde en client léger. Cela améliore l'accès à l'application à un plus grand nombre de personnes mais aussi la maintenance de ce type d'application.

Les chapitres suivants vont compléter cette présentation en offrant une description détaillée des différentes solutions logicielles et matérielles existantes pour la mise en place de l'architecture.

Je découpe cette présentation en deux modules principaux :

- Système de gestion de données
- Système de gestion des applications

4.2 Système de gestion des données

4.2.1 Présentation

La base de données est un système de stockage qui met à disposition des données pour tous les types de clients applicatifs.

Actuellement, il existe déjà un SGBD dans notre architecture. Ce système avait été créé suite à l'obsolescence d'une ancienne base de données sous Informix. La mise en place de ce SGBD a été réalisée rapidement, il n'y a pas eu de conception ce qui fait que les tables n'ont aucune correspondance entre elles. Avec l'augmentation du nombre et de types d'informations à stocker, il n'est plus possible de récupérer et traiter correctement l'information. La recherche et la fourniture des données ne correspondent plus au besoin actuel demandé par la RATP.
De plus, ce système de données est composé d'un seul serveur. En cas de panne, le SGBD n'est plus en mesure de fournir le besoin.

Pour pallier à ces problèmes, l'architecture matérielle et logicielle de la base de données doit être revue entièrement pour améliorer la disponibilité du système ainsi que la fourniture des données.

Les lignes de RER et du Métro sont composées d'un certain nombre d'éléments pouvant être reliés entre eux. La réalisation d'un modèle relationnel devient obligatoire.

La conception est réalisée en deux étapes :

- Modèle conceptuel de données permettant de définir la structure du système au niveau des données mais aussi de connaître leurs relations entres elles.
- Modèle physique de données permettant de traduire ou d'implémenter le modèle conceptuel dans le SGBD. Pour notre cas, nous utiliserons le langage Structured Query Language (SQL).

Pour la mise en relation des données entre elles, l'utilisation de la spécification fonctionnelle de l'exploitation nous permet de connaître en détail chaque objet/équipement d'une ligne. La conception s'appuie aussi sur les données déjà existantes récupérées sur les différentes applications.

Ce système a pour but de remplacer une base existante. L'intégration de ce système ne doit en aucun cas engendrer de dysfonctionnement sur l'exploitation. Une intégration par étape est nécessaire pour s'assurer de son bon fonctionnement du fait qu'un certain nombre d'applications annexes ne sont pas encore portées pour fonctionner sur le nouveau système de base de données.

4.2.2 Analyse des solutions

4.2.2.1 La méthodologie

Le choix d'architecture d'une base de données dépend principalement du besoin que l'on souhaite en faire. Pour comprendre le choix technique retenu, une étude est faite sur les différentes structures possibles pour la réalisation de cette architecture.

Pour des raisons de préconisation RATP, certains logiciels me sont imposés. Ils ne représentent pas une contrainte en termes de performance. Ils ont été choisis suite à leurs installations sur différents projets validés par la RATP.

Pour la partie matérielle, nous utiliserons des serveurs de DELL avec un système d'exploitation Linux CentOS 6.x. La préconisation se positionne sur cet OS.
Pour la partie du SGBD, nous utiliserons le moteur de base de données PostgreSQL.

Actuellement, la base de données utilise le moteur MySQL. Elle pourrait convenir pour notre future architecture, cependant nous souhaitons uniformiser nos logiciels pour nous concentrer sur une réduction de la maintenance applicative mais aussi sur l'administration de nos systèmes.

Les besoins attendus sont multiples. Pour la réalisation d'un système performant, Jérôme Poussineau explique qu'il faut répondre à plusieurs critères **[11]** :

- La disponibilité du système (*availability*[5])
- Les temps de réponse (respond time[6])
- La montée en charge (scalability[7])
- La répartition de charge

[5] Availability : disponibilité
[6] Respond time : les temps de réponse
[7] Scalability : évolutivité

4.2.2.2 La disponibilité du système

La disponibilité d'une base de données correspond au temps de fonctionnement d'un système avant une panne **[12]**.

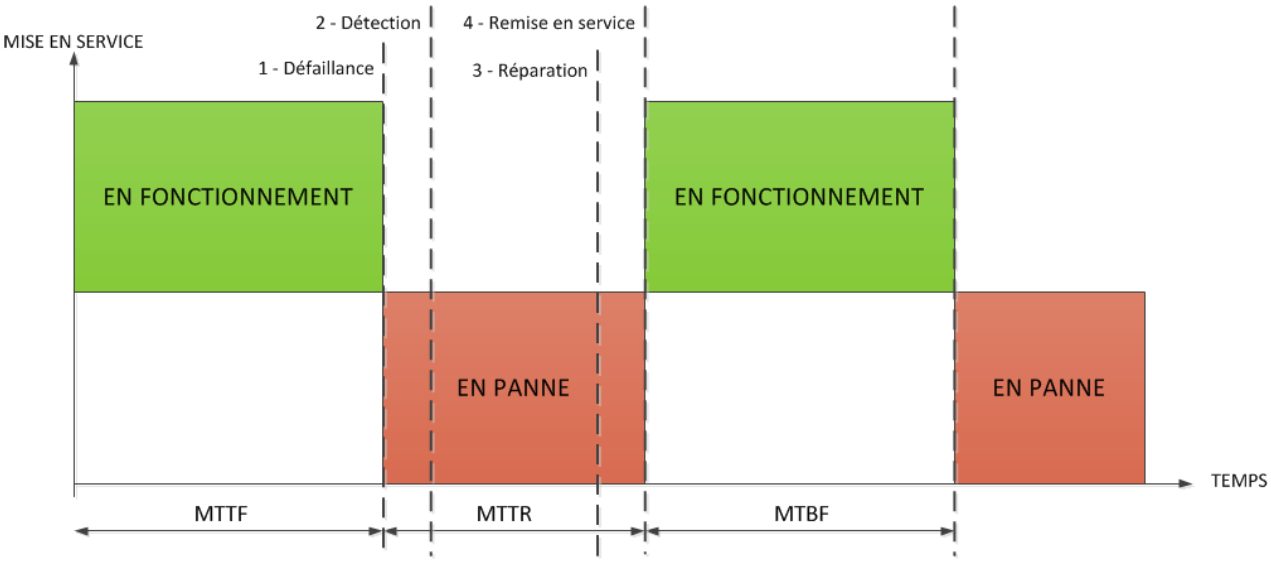


Figure 18 - Fonctionnement de la disponibilité

MTTF (*Mean Time To Failure*) : Temps moyen de fonctionnement avant panne
MTBF (*Mean Time Between Failure***) :** Temps moyen entre pannes
MTTR (*Mean Time To Repair***) :** Durée moyenne de réparation

C'est un critère important pour le choix de notre architecture. Elle peut être gérée de deux manières, cela dépend du besoin que l'on souhaite mettre en place.

La première proposition est de prévoir une architecture où la disponibilité atteint les 100%. C'est de la haute disponibilité. Il faut prévoir un MTTF qui soit le plus long possible.

La deuxième proposition est de prévoir un système ayant une disponibilité inférieure à 100%. Ce système est moins contraignant. Il autorise l'arrêt de la base de données. L'avantage d'un arrêt volontaire nous permet d'effectuer une maintenance applicative pour réaliser des sauvegardes ou des mises à jour de la base de données. Cependant en cas d'arrêt, pour la récupération d'information il faut réaliser des scripts de réinsertion de données.

Notre besoin consiste à ne perdre aucune information et d'être disponible pendant toute la période d'exploitation. Sa durée peut-être de 24h lors de certaines journées spécifiques.

J’ai donc choisi de proposer un système avec une haute disponibilité. De plus, certaines applications ont pour objectif de fournir les informations du suivi temps réel vers les applications d’affichage. Une réinsertion de données viendrait perturber le fonctionnement des applications.

Dans nos exigences de maintenance, il est défini qu’en cas de panne, nous devons intervenir pour les applications de suivi temps réel dans un délai de 30 minutes. Ce qui signifie qu’avec ce système, en cas de panne de l’un des serveurs, nous améliorons les conditions d’interventions de maintenance qui soumettent le mainteneur a une certaine pression.

4.2.2.3 Les temps de réponse

C'est un critère, qui correspond au temps que doit fournir un système dès lors qu'il reçoit une demande ou une commande. Pour une base de données, cela revient à fournir des informations dans un temps imparti.

Dans nos applications de suivi et d'aide à l'exploitation, l'accès à la base de données est permanent en écriture pour archiver l'ensemble des informations du suivi. L'accès pour la récupération des données est moins fréquent. Elle se fait lorsque l'offre de transport est modifiée. C'est une action habituelle en début de journée d'exploitation mais réalisable en cours de journée par l'intervention du régulateur.

Ces opérations doivent être rapidement effectuées. La spécification autorise un délai maximum de deux secondes pour le temps de réponse du faite qu’un nombre important de données est à récupérer. Pour la nouvelle base, nous restons sur ces mêmes règles d’exigences.

D’après un article de Frédéric Brouard[8] **[13]**, le temps de réponse d’une base dépend de la manière dont elle a été conçue et suivant le nombre d’éléments que la base contient.
Lorsqu’une base de données possède peu de tables avec beaucoup de données, le temps de réponse devient de plus en plus long. Inversement, lorsque l’on se retrouve avec plus de tables qui utilisent de bonne relation entre elles, la récupération des données ce fait plus rapidement lorsqu’il y a beaucoup de données, ce qui diminue le temps de réponse lors de traitement.

[8] **Frédéric BROUARD** : Expert du langage SQL et des bases de données relationnelles. Il a réalisé le site **SQLpro** hébergé par **www.developpez.com**. C'est un spécialiste MS SQL Server, reconnu à titre de **MVP** par Microsoft.

4.2.2.4 La montée en charge

La montée en charge correspond à une période où l'on sollicite très fortement le réseau par une augmentation du nombre de connexions simultanées. Sa mauvaise gestion occasionne certains risques sur le réseau. Le plus souvent, il apparaît sous forme d'un fort ralentissement. Ce risque peut engendrer des pannes en cascade.

Nos processus de communication effectuent des demandes de présence cyclique avec les applications en connexion. Si l'une des applications ne répond pas dans un certain délai, on la déconnecte. La coupure de connexion rompt la chaîne de traitement avec une perte d'information du suivi.

Les principales requêtes faites sur les serveurs sont de type TCP, HTTP et SQL :

- Les requêtes TCP représentent la communication entre le serveur de suivi et le serveur d'application.
- Les requêtes HTTP sont utilisées pour l'appel des services web.
- Les requêtes SQL sont exécutées par les services web pour accéder aux données sur la base de données.

La montée en charge peut être contrôlée. Pour cela, les serveurs doivent subir des tests de charge pour définir leurs configurations et leurs limites applicatives.

Il existe plusieurs applications venant du libre pour la réalisation de ces tests. Après recherche, je souhaite utiliser une application multi-protocole appelée « **Tsung** [9]». Elle permet aussi la lecture de scénario ce qui automatise les tests. Notre besoin est de vérifier si les bases de données ainsi que nos serveurs d'applications peuvent tenir en cas de forte affluence sur le réseau.

Un outil de rejeu a été conçu pour isoler la plateforme de tests. Il permet de rejouer les messages de suivi des trains que l'on a archivés sous forme de fichier. Nous avons la possibilité d'exécuter plusieurs instances de ce processus pour un test avec plusieurs connexions. Actuellement, il n'existe que deux connexions TCP pour deux messages.

[9] TSUNG : Outil open source multi-protocole pour la réalisation de tests de montée en charge - http://tsung.erlang-projects.org/

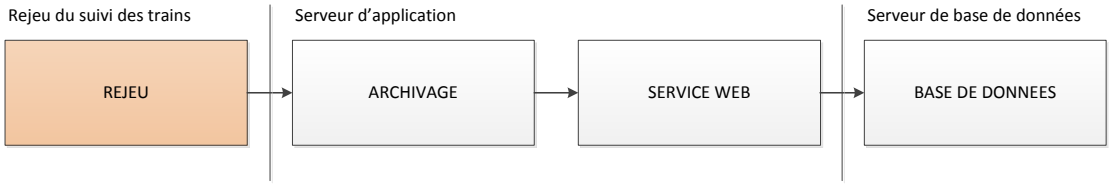


Figure 19 - Position dans l'architecture du processus de REJEU

Dans la finalité de ce projet, il n'existera plus ce type de connexion. Un des avantages de cet outil est qu'il se positionne en début de chaîne de l'architecture. Nous pouvons tester son ensemble en jouant sur la vitesse de lecture.

4.2.2.5 La répartition de charge

La répartition de charge **[14]** peut-être gérée par l'intermédiaire d'une architecture adaptée. La première proposition d'architecture consiste à mettre en parallèle deux bases de données dont l'une sert à l'insertion des données et la seconde à la lecture.

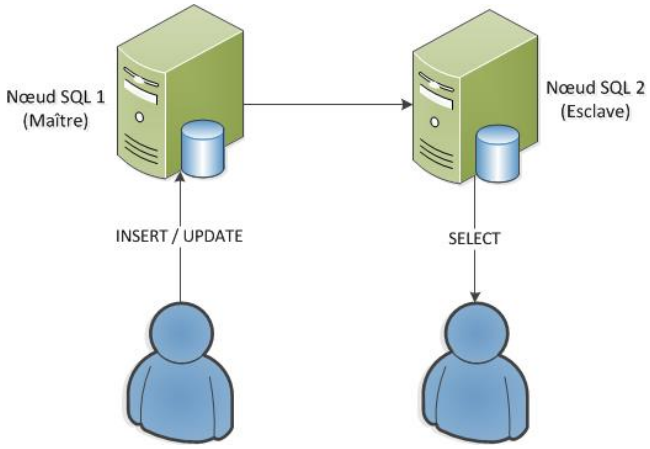


Figure 20 - Serveur répliqué : un serveur en écriture et un serveur en lecture

Le fonctionnement nominal de ce système permet de séparer les tâches de traitement d'une base à une autre. Lorsqu'une panne apparaît sur l'une, la base de données encore active prend le relais de la fonctionnalité manquante.

L'inconvénient de ce système est qu'il nécessite de maintenir deux machines en fonctionnement. Le principe de répartition de charge appliqué est rompu lorsque l'une des deux machines tombe en panne.

Nous avons voulu reprendre ce principe sur notre base actuelle. Cela sépare les accès internes dédiés aux exploitants et les accès externes accessibles uniquement en lecture via l'intranet de la RATP. Une erreur a été commise dans sa mise en place. Lorsque la base de

données principale s'interrompt, aucun système ne reprend la main. Le serveur secondaire n'est plus mis à jour. L'insertion des données sur le secondaire n'étant pas possible.

Une seconde architecture est possible. On utilise une ferme de serveurs plus connu sous le nom de « cluster de serveurs ». Il est conçu pour permettre l'accès à ces serveurs par une seule et même adresse IP. On a la possibilité d'installer autant de serveurs que l'on souhaite avec un minimum de deux.

L'avantage par rapport à la première architecture, c'est qu'elle permet l'utilisation de tous les serveurs en lecture. De plus, lorsqu'une base de données tombe en panne, une autre reprend la main en tant que maître pour l'insertion en mode synchrone.

Avec cette méthode, on permet une répartition de la charge sur plusieurs serveurs. On nomme cette étape « Load balancing [10]»

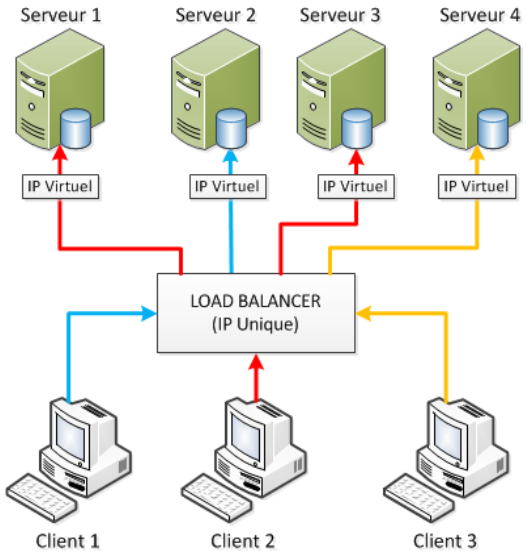


Figure 21 - Architecture pour le load balancing

Le moteur de base PostgreSQL ayant été choisi pour la réalisation de ce projet pour suivre les directives de la préconisation. PostgreSQL nous propose d'étudier différentes méthodes qu'il est possible d'appliquer avec leur moteur **[4]**.

Une des méthodes décrites correspond à l'utilisation d'un « middleware » nommé « Pgpool II ». Elle réalise l'interface entre le client et la base de données.

[10] **Load balancing** : répartition de charge

Cet outil permet la réalisation d'un ensemble de fonctionnalités :

- réplication de données
- répartition de charge : load balancing
- watchdog entre processus (signe de vie)
- resynchronisation des données en cas de crash

Cet outil permet d'ajouter des procédures automatiques de synchronisation. Ce qui évite maintenant d'arrêter l'ensemble du système de base et la perte éventuelle de données. Cette méthode est appelé « failback ».

Cette méthode a déjà été mise en place sur un de nos projets. C'est un projet qui a quelques années, les versions de « PostgreSQL » et de « Pgpool II » utilisées ne permettait pas la gestion de resynchronisation. Sur ce projet, la resynchronisation a été prévue différemment. En cas d'arrêt de l'un des deux serveurs, il faut resynchroniser manuellement les données avant de le relancer à partir d'un dump de la base secondaire. Si elle n'est pas faite, Pgpool II fait une vérification d'équivalence des bases de données. On se retrouve avec des bases désynchronisées. L'ensemble du système s'interrompt. De plus, la vérification des processus (watchdog) dans ce projet n'e pas été gérée par Pgpool II mais par un applicatif externe venant du libre qui se nomme Ucarp.

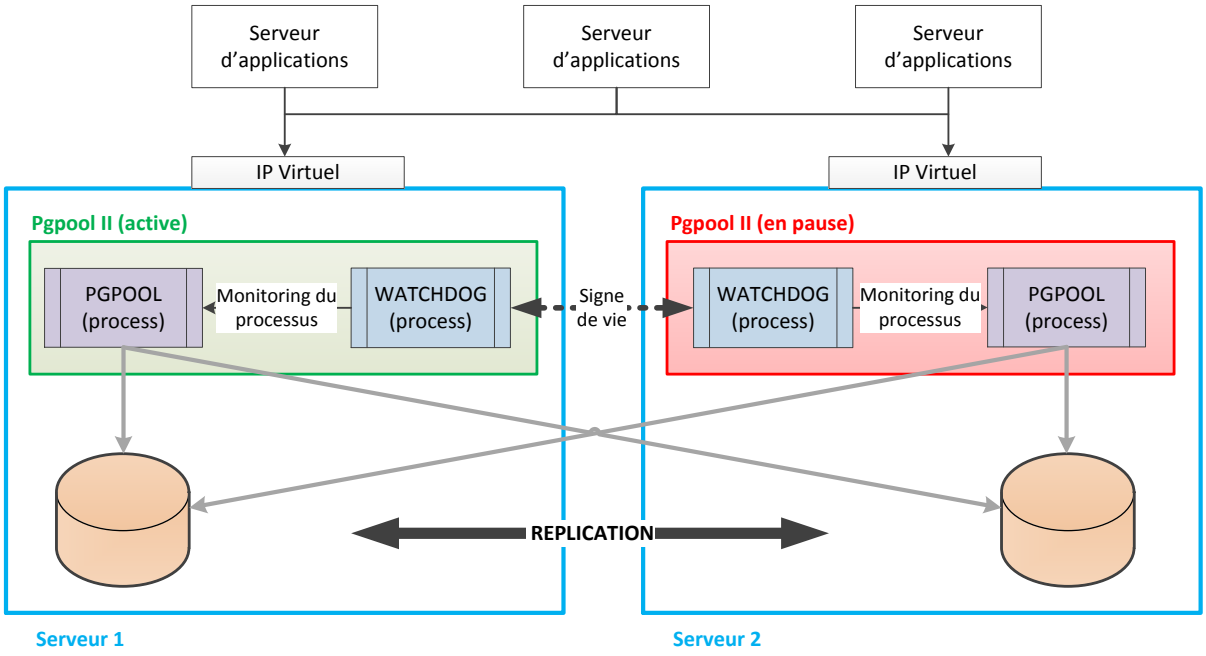


Figure 22 - Fonctionnement de Pgpool II

Le signe de vie entre les processus Watchdog de chaque serveur leurs permettent de gérer la réplication, la redondance en cas de panne de l'un des deux et la répartition de charge sur l'ensemble des bases.

L'avantage d'un « middleware », c'est qu'il ne nécessite pas de matériels spécifiques pour fonctionner. Venant du libre, aucune licence n'est à prévoir.

L'article PostgreSQL présente aussi une méthode avec partage d'un disque partagé, deux méthodes multi-maîtres et une réplication avec des triggers qui revient à faire une insertion asynchrone. Elles ne correspondent pas à notre besoin de disponibilité et une mise en temps réel des données.

La dernière méthode présentée est plus complexe à mettre en place. Nous sommes un service de maintenance applicatif et non des experts en base de données. Il est important de proposer des systèmes qui soient performants et facilement pris en main. Plus le système sera complexe et plus il nécessitera de s'approprier des compétences spécifiques pour la maintenance applicative.

Les méthodes de redondance présentées sont de type logiciel. Il existe des solutions de type matériel. Ces méthodes bien que très performantes sont très coûteuses à mettre en œuvre. On considère qu'il est parfois plus judicieux de maintenir plusieurs machines en cluster qu'une seule machine spécialisée **[15]**.

4.2.3 Architecture logicielle de la base de données

Voulant proposer un système ouvert à tous les types de clients applicatifs, le choix de la base de données se dirige vers un système où la performance doit être optimale.

Notre besoin se porte sur un accès en lecture et écriture. L'accès en temps réel des données sur tous les serveurs est indispensable pour la réalisation d'une bonne répartition de charge. D'où l'abandon de la méthode asynchrone.

Mon choix se porte sur un système maître-esclave avec une réplication de données/insertion de type synchrone permettant l'intégrité des données (*cf chapitre **2.3.3.2.2***).

Pour la répartition de charge, mon choix se porte sur l'utilisation du middleware PGPool II (*cf Figure 22 - Fonctionnement de Pgpool II*). Il a pour avantage aussi de gérer la méthode de réplication maître-esclave de type synchrone. C'est un outil performant, qui réalise et simplifie la mise en place de la réplication des données et de la répartition de charge.

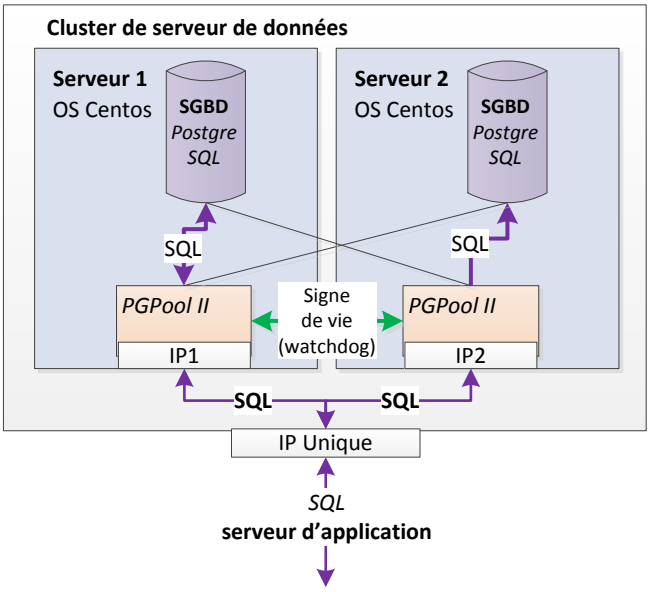


Figure 23 - Architecture logicielle du cluster de la base de données

Pour simplifier la présentation de l'architecture logicielle, je les ai séparées volontairement pour la présentation (*cf chapitre **4.4***).

4.2.4 Architecture matérielle de la base de données

La disponibilité est l'élément le plus important. Le cluster de serveurs représente la meilleure solution en termes de structure.

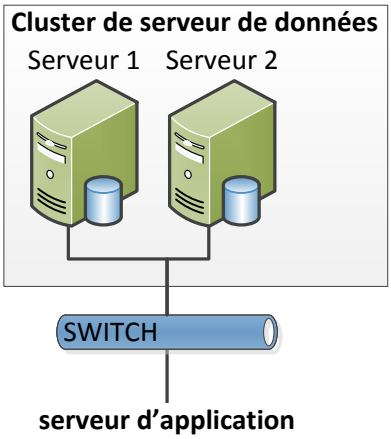


Figure 24 - Architecture matérielle du cluster des serveurs de la base de données

Pour simplifier la présentation de l'architecture logicielle, je les ai séparées volontairement pour la présentation (*cf chapitre **4.4***).

4.2.5 Conception de la base de données

L'une des tâches les plus complexes a été de concevoir un nouveau modèle de données. Plusieurs facteurs rentrent en ligne de compte pour cette conception.

La base de données ne doit pas provoquer de modifications externes. Il faut qu'elle soit capable de fournir les fichiers de données actuels pour chacune des applications existantes. L'objectif principal est de supprimer tous les fichiers de configuration interne à chacune des applications.

Pour la conception de cette base de données, il ressort trois grandes familles :

- les données d'archivages
- les données topologiques
- les données de l'offre de transport

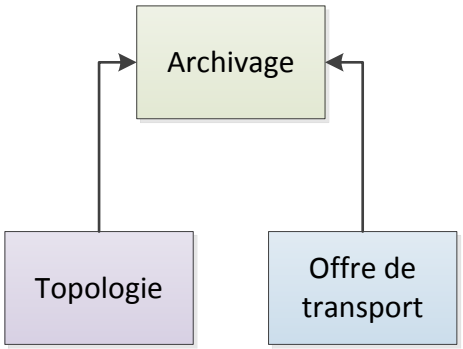


Figure 25 - Conception de la base de données : les 3 grandes familles

Les données d’archivage proviennent du suivi des trains. Nous complétons ces informations en leurs associant des données de l’offre de transport de la topologie.

4.2.5.1 Les données d’archivages

Sur la base de données actuelle, nous avons réalisé une table par événement. Il manque dans cette méthode une table de liaison pour obtenir un fil de l'eau général de tous les événements.

On se retrouve avec des tables ayant des événements avec une même date sans connaître leurs ordres d'arrivées. Pour nous rattraper, nous avons mis en place des dates d'insertions automatiques à la milliseconde avec l'attribut SQL « TIMESTAMP ».

Pour cette conception, afin de ne pas reproduire les erreurs sur l’ordre d’arrivée des événements, nous avons ajouté une table d'archive générale qui regroupe l'ensemble des informations au fil de l'eau. On fait la différence entre une date d’insertion et une date d’événement. Cette méthode permet de réinsérer des données.

Cette table est en liaison avec des tables spécifiques. Elles permettent un découpage par fonctionnalité. Nous recevons actuellement deux messages. Ce sont les numéros 099 et 100. Seul le message 100 a été mis dans le modèle et découpé par fonctionnalité. Le message 099 reprendra le même principe de conception.

4.2.5.1.1 MSG099 : Le message de suivi et de gestion d'exploitation

C'est un message générique contenant différents types d'éléments. Il permet la remontée de toutes les informations du suivi des trains et de la gestion d'exploitation. C'est un message qui a été créé spécifiquement pour une insertion en base de données. Ce message remplace le message 097 qui alimentait une base de données Informix.

C'est un message au fil de l'eau non événementiel, il est envoyé lorsqu'un nombre de vingt événements a été stocké en mémoire. Il ne sera pas traité dans ce mémoire mais ultérieurement.

4.2.5.1.2 MSG100 : Le message de suivi des trains

Il correspond au message de suivi des trains. Le MSG099 permet un suivi moins détaillé. C'est un message événementiel. Il permet l'envoi de cinq types d'événements. Les événements sont envoyés séparément. Dans le même principe que le message 099, c'est un message qui remplace le message 186 pour une insertion en base.

Type de l'événement	Intitulé du message	Rôle du message
H	Horaire	Donne le numéro et l'indice du graphique horaire pour sélectionner les fichiers trains et dessertes
D	Départ	Donne l'heure de départ de gare de la mission
A	Arrivée	Donne l'heure d'arrivée en gare de la mission
O	Occupation CDV	Indique sur l'occupation d'un circuit de voie
L	Libération CDV	Indique sur la libération d'un circuit de voie

Tableau 2 - Evénement de l'archivage du suivi des trains

D'après la spécification d'interface, on peut regrouper certains événements qui remontent les mêmes informations.

Les événements d'arrivée et de départ sont regroupés dans une seule table. Du fait qu'il y a que le type d'événement qui change. De même, les événements d'occupation et de libération sont regroupés dans une seule table. Seul le type d'événement change.
Pour ces deux tables, elles ont des relations avec l'offre de transport (OFT) et la topologie (TOP) (cf ***Annexe 1 : Modélisation conceptuelle des tables de l'archivage***).

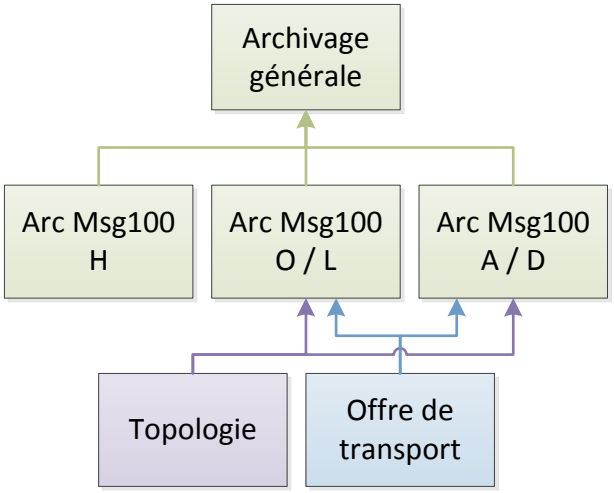


Figure 26 – Modélisation des tables d'archivage du MSG100

L'événement du graphique horaire est unique. Il y a une relation avec la table de l'offre de transport. Après chaque chargement de graphique, on récupère les informations associées dans nos applications pour limiter l'accès à la base en cas de besoin.

J'ai choisi de traiter ce message plutôt que le message 099 pour plusieurs raisons.
Le message 099 est déjà traité par une application web pour un désarchivage d'informations.

Avec le message 100, j'ai la possibilité de traiter deux sujets :

- portage de client lourd en client léger
- suppression des redondances

Une application de graphiquage de type client lourd existe actuellement. Elle utilise le message 186. Pour limité les redondances d'informations sur le réseau, on porte l'application sous forme de client léger et on utilise le message 100.

4.2.5.2 Les données topologiques

On peut définir chaque élément de la ligne, on appelle ces données la topologie. Une étude a permis de concevoir les différents liens entre chaque équipement/élément de la ligne.

Cette conception se base sur deux documents principaux que sont la spécification fonctionnelle de la ligne du RER A et un plan détaillé de l'ensemble de la ligne du RERA.

Ces données existent actuellement sous forme de fichier et sont utilisées par les serveurs de suivi. Faire abstraction de ses fichiers a été difficile mais nécessaire pour concevoir notre système. Ces fichiers de données seront utiles pour remplir la nouvelle base de données dans un premier temps.

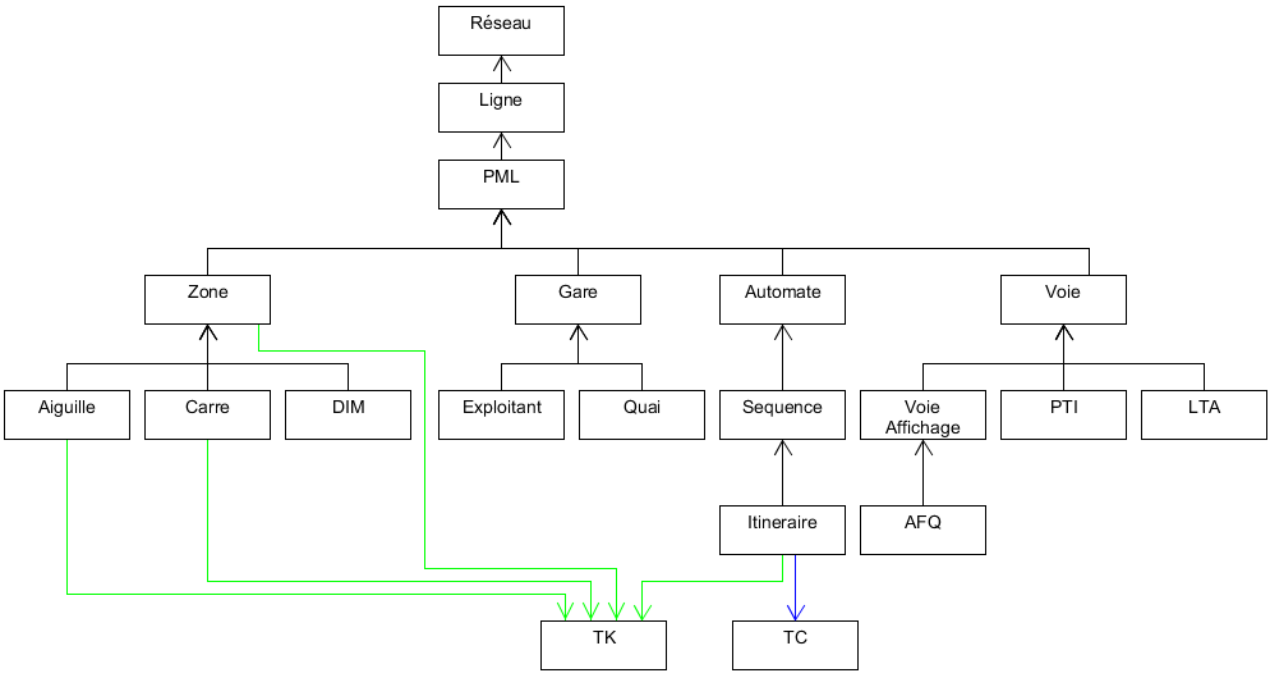


Figure 27 - Modélisation complète des objets de la topologie

J'ai réalisé cette modélisation suivant une hiérarchie de l'élément le plus général au plus précis. Il y a plusieurs types d'éléments. J'ai découpé la présentation de la modélisation en 2 parties dans les annexes 2 et 3 (*cf* ***Annexe 2 : Modélisation conceptuelle de la topologie (1/2)*** *et* ***Annexe 3 : Modélisation conceptuelle de la topologie (2/2)***).

Le premier type d'éléments permettant de décrire la ligne :

- « Réseau » : RER, Métro et Tramway
- « Ligne » : A, B, 1, 2, 3, 3bis, 4, 5, …etc

- « Exploitant » : RATP, SNCF, …

Le second type d'éléments permettant de décrire chaque équipement de la ligne :

- PML (Poste de Manœuvre Local) : Poste permettant de gérer en local un secteur de la ligne. Il gère un ensemble de gare.
- Gare : Élément composé de plusieurs quais.
- Quai
- Voie
- Zone : Ce sont l'ensemble des circuits de voie de la ligne.
- Carré
- Aiguille : Ce sont des équipements de guidage permettant de modifier la destination d'un train suivant son itinéraire.
- DIM
- PTI (Point d'Identification)
- LTA (Listes des Trains)
- AFQ (Equipement d'affichage)

Les derniers éléments permettent de gérer les circulations sur la ligne, ce sont des éléments fortement liées aux équipements :

- Automate
- Séquence
- Itinéraire
- TC (Télécommande) : Elément permettant de commander les itinéraires sur le terrain.
- TK (Télécontrôle) : Elément permettant de faire remonter l'information de changement d'état des équipements de zone, carré, aiguille et itinéraire.

4.2.5.3 Les données de l'offre de transport

Les données de l'offre de transport ne sont pas gérées par notre service. C'est le département MTS/HORAIRE qui nous les fournit. Une refonte de l'offre de transport en termes de données et de format est en cours de réalisation. Elle devrait être mise en production en 2016.

Cependant, il est obligatoire de relier à l'offre actuelle à notre architecture. La conception a donc été simplifiée. (*cf* ***Annexe 4 : Modélisation conceptuelle de l'offre de transport***)

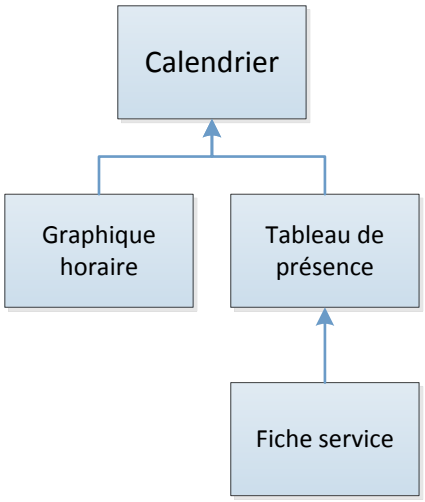


Figure 28 - Modélisation de l'offre de transport

Le calendrier d'application est validé chaque année par le STIF. Il indique pour chaque jour de l'année, le graphique horaire et le tableau de présence à appliqué. Suivant certaines situations d'exploitation, le régulateur a la possibilité de choisir un graphique horaire pour palier à des anomalies sur la ligne.

Les graphiques horaires représentent une journée d'exploitation et peut-être est appliqué pendant plusieurs semaines. De part un nombre important d'informations pour le suivi, ils ont découpé les informations en plusieurs fichiers.

Le premier fichier est appelé « COURSES ». Il indique un ensemble d'informations sur chacune des circulations :

- Heure de départ en terminus
- Heure d'arrivée en terminus
- Voie de départ
- Voie d'arrivée

- Nom de mission
- Type de circulation
- Type de matériel

Le second fichier est appelé « TEMPS ». Il indique l'heure d'arrivée et de départ pour chacune des gares traversées par une circulation.

Le dernier fichier est appelé « POINTS ». Il permet d'indiquer des informations suivant un point de la ligne. Un point pouvant être l'élément GARE. On en récupère les informations de temps de stationnement et de temps de parcours pour une circulation.

Les tableaux de présence contiennent toutes les informations liées aux fiches services. Une fiche service décrit pour un conducteur sa feuille de route.

Les tableaux de présence contiennent toutes les informations liées aux fiches services. Une fiche service décrit pour un conducteur sa feuille de route.

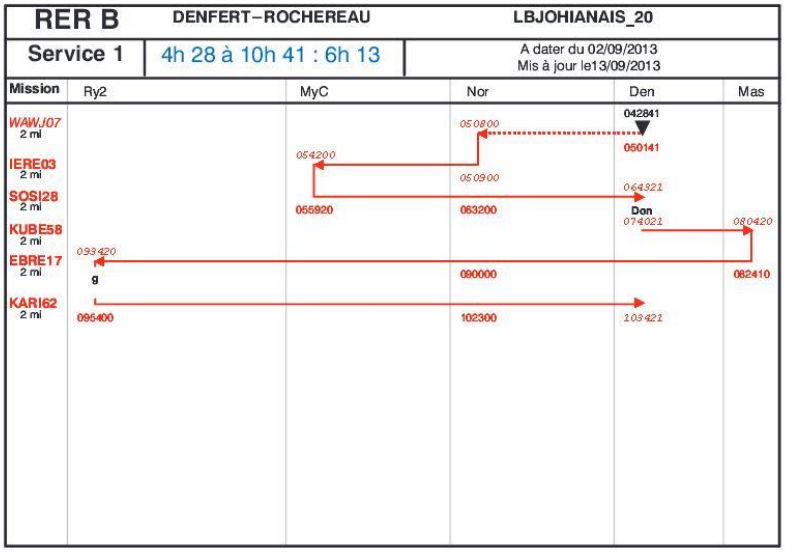


Figure 29 - Image d'une fiche de service d'un conducteur sur le RERB

L'ensemble de l'offre de transport subit actuellement une refonte de la part du département RER. Les services conducteurs n'étant pas en liaison avec le système d'archivage, j'ai choisi de ne pas le mettre en place dans mon modèle. Nous ne connaissons pas encore les données et le format utilisé pour la suite.

4.2.5.4 Méthodologie de conception

Pour la réalisation du modèle conceptuel et du modèle physique, nous avons utilisé un outil. C'est un outil distribué par SAP du nom de Sybase PowerAMC.

Pour la nouvelle structure, nous utilisons les clés étrangères. Cela apporte une intégrité des données dans chacune des tables en relation. Cette méthode n'existe pas sur la base actuelle.

Les données d'archivage sont stockées sur une période glissante de trois mois.

Les données topologiques ne doivent jamais disparaître de la base de données. Même dans le cas d'une modification ou d'une suppression, il est nécessaire de sauvegarder une trace. Il a été prévu pour chaque élément, d'ajouter des attributs de temps pour une vérification de validité des données. Les graphiques horaires étant uniques, en cas de modifications, le département des horaires nous remettra une nouvelle offre de transport.

Voici un tableau des éléments de temps associés à chacun des objets :

Date d'insertion	utilisé pour connaître sa date d'insertion dans la base. Cet attribut est généré automatiquement par le moteur de base de données.
Date d'événement	utilisé uniquement pour les archives de suivi, cela permet de faire une différence entre l'insertion en base et la véritable date de l'événement.
Date de début	utilisé pour connaître sa date de début de validité en relation avec le suivi des trains.
Date de fin	utilisé pour connaître sa date de fin de validité pour en relation avec le suivi des trains.

Tableau 3 - Argument de DATE pour les objets topologiques

Avec ce système, un objet peut-être en doublon dans le cas où il est modifié. Une suppression n'engendre pas un nouvel ajout.

Dans la conception, j'ai choisi de ne pas séparer les dates d'un élément. Ce choix se porte sur une meilleure lecture de la base de données. Il vient aussi du fait qu'une donnée topologique est rarement modifiée mais plutôt ajoutée par un nouvel identifiant. D'après

l'historique, une version des données topologiques est réalisée tous les ans avec principalement des ajouts et des suppressions.

Dans le cas où cela se produit, les applications doivent pouvoir retrouver l'objet sans faire de traitement de date. Pour ne pas ralentir le système en cas de forte charge, nous avons rajouté un champ de type booléen nommé « xx_FLAG ». Cela permet aux applications de se positionner directement sur l'objet à utiliser. Le flag est modifié lors de la modification ou de la suppression d'un élément.

Prenons un exemple avec l'objet zone « Z1037 » en cas de modification. En cas de suppression il n'y aura pas la seconde ligne.

nom	date d'insertion	date de début	date de fin	flag
Z1037	13/04/2015	14/04/2015	19/04/2015	0
Z1037	22/04/2015	20/04/2015	-	1

Tableau 4 - Exemple des éléments DATE lors de la modification d'un objet ZONE

4.2.6 Récupération et intégration de la base de données

Une opération délicate consiste à récupérer les informations de la topologie des lignes du RER et du Métro pour les insérer dans le nouveau format de la base de données.

Une contrainte importante arrive rapidement lorsque l'on s'aperçoit que les données de chaque ligne ne se présentent pas de la même manière que ce soit au niveau du format mais aussi des attributs de chaque élément. En réalisant une étude de comparaison, on retrouve les informations et leurs liaisons ce qui permet de compléter la base. On décide de se concentrer sur la ligne A du RER. Les données de la ligne B puis celles du Métro seront traitées naturellement après.

Actuellement, les données topologiques de la ligne du RER A sont générées par un outil sous forme de fichier. Cet outil de génération de données est un programme écrit en C et compilé sous Solaris. Il est relativement complexe dans sa méthode de vérification de cohérence entre toutes les données. Chaque fichier représente la liste de toutes les données d'un type d'objet de la topologie. Il existe une trentaine d'objet.

Cette application est en prévision de portage sous forme de client léger. Elle accédera à la base de données via les services web pour ajouter, modifier et supprimer les données. Elle ne sera pas présentée dans ce mémoire pour des raisons de temps de réalisation.
L'outil n'étant pas commencé, il est important de trouver un système temporaire et rapide pour insérer les données pour tester l'ensemble du système.

L'insertion par scripts est bien entendu une étape temporaire dans le déroulement du projet. Dans le souhait que l'application web soit opérationnelle rapidement pour configurer et mettre à disposition les données.

Les fichiers de données n'utilisant pas un format normé tel que le comma-separated values (CSV) ou l'extensible markup language (XML) il m'était impossible de les insérer directement dans le format de la nouvelle base de données.

La première étape correspond à insérer les données de topologie dans la nouvelle structure de la base de données.

La technique employée est de créer dynamiquement des tables temporaires de tous les objets avec leurs attributs et leurs données associées par l'intermédiaire d'un script.

Les attributs permettent de créer la table d'un objet et les données permettent de la remplir.
Le script a été écrit dans le langage PERL. Ce langage est connu pour sa simplicité à réaliser du traitement de chaîne de caractères.

Prenons l'exemple des objets « équipements d'affichage » (AFQ) qui sont composés de quatre attributs, ils sont représentés dans le fichier sous cette forme :

```
Equipement_d'affichage => debut
  Nom => "GER-SI"
  PML_d'appartenance => "GER"
  Numero_de_l'equipement_d'affichage => 0
  Liste_associee => "GER03"
Equipement_d'affichage => fin
```

Figure 30 - Exemple objet de la topologie : équipement d'affichage

Le libellé en rouge correspond à la délimitation de l'objet. En vert ce sont les attributs et en bleu les données.

Après lecture du fichier, on réalise la requête de création de la table temporaire au format SQL que l'on insère dans un fichier nommé « nomdelobjet_TABLE.sql ». Les requêtes pour l'insertion de données seront dans un nouveau fichier nommé « nomdelobjet _DONNEES.sql ».

Les fichiers pour la création des tables temporaires ne sont créés qu'une seule fois. Il peut être régénéré lorsque l'on modifie la structure de la base de données.
Les fichiers d'insertion de données sont quant à eux générés à chaque modification des données.

Une seconde étape consiste à insérer les données de la base actuelle vers la nouvelle base. Les données contenues dans la base actuelle sont des données d'archivage du suivi des trains. Un avantage dans cet exercice vient du fait que les données actuelles ne sont sauvegardées qu'uniquement sur trois mois glissant.

Le choix le plus pertinent qui a été trouvé est de mettre en parallèle la base de données existante les deux bases de données. Cette technique nécessite néanmoins de modifier la configuration de communication des serveurs de suivi des trains.

La migration de l'ancien vers le nouveau système de données est détaillée dans un chapitre suivant « la migration » (*cf chapitre **4.6***).

4.3 Système de gestion des applications

4.3.1 Présentation

Un serveur d'application va être utilisé pour centraliser l'accès à la base de données. Cela permet d'assurer un contrôle d'accès, d'assurer un contrôle des données insérées et d'alléger la base de données de tous les traitements pouvant perturber son fonctionnement.

Sur la base existante, les traitements sont conçus sous forme de scripts exécutés quotidiennement pour la génération de rapport (ex : fichier kilométrique des matériels sur une journée) ou de processus d'archivage. Ces traitements utilisent directement sur le serveur de base de données. Dans le nouveau système, ils sont tous déplacés pour être hébergés par le serveur d'application.

Pour simplifier et unifier le type d'accès à la base de données, ce serveur est également voué à accueillir un serveur web pour la réalisation de services Web.

4.3.2 Architecture logicielle de la gestion des applications

Pour la disponibilité, on prend le même schéma que la base de données avec un cluster de serveurs. Il existe cependant une différence avec le cluster de la base de données. Elle vient de la gestion de la redondance qui est réalisé par une solution alternative qui est UCarp. C'est une application que j'ai découverte sur l'une de nos applications maintenues dans notre entité.

UCarp est un outil venant du libre permettant la mise en place d'un cluster de serveur pour de la haute disponibilité. C'est un outil qui s'installe sur chaque serveur du cluster. L'élément UCarp va communiquer avec les autres UCarp par l'intermédiaire d'un signe de vie. Cet outil ne permet pas de la répartition de charge.

UCarp fonctionne sur le principe du maître-esclave. En cas de panne de l'un des serveurs, l'un des autres UCarp du cluster prend la main en tant que « Maître ». Une fois le serveur redémarré, il faut relancer l'outil UCarp pour réintégrer le serveur au cluster.

Pour les processus présents sur le serveur (archivage, serveur web, etc…), il est nécessaire de réalisé un processus espion pour surveiller ces processus. En cas de défaillance de l'un des processus, l'espion prévient UCarp du dysfonctionnement pour qu'UCarp passe la main

à un autre serveur. La relance n'impose pas une synchronisation des données du fait que les applications ne gardent aucun état.

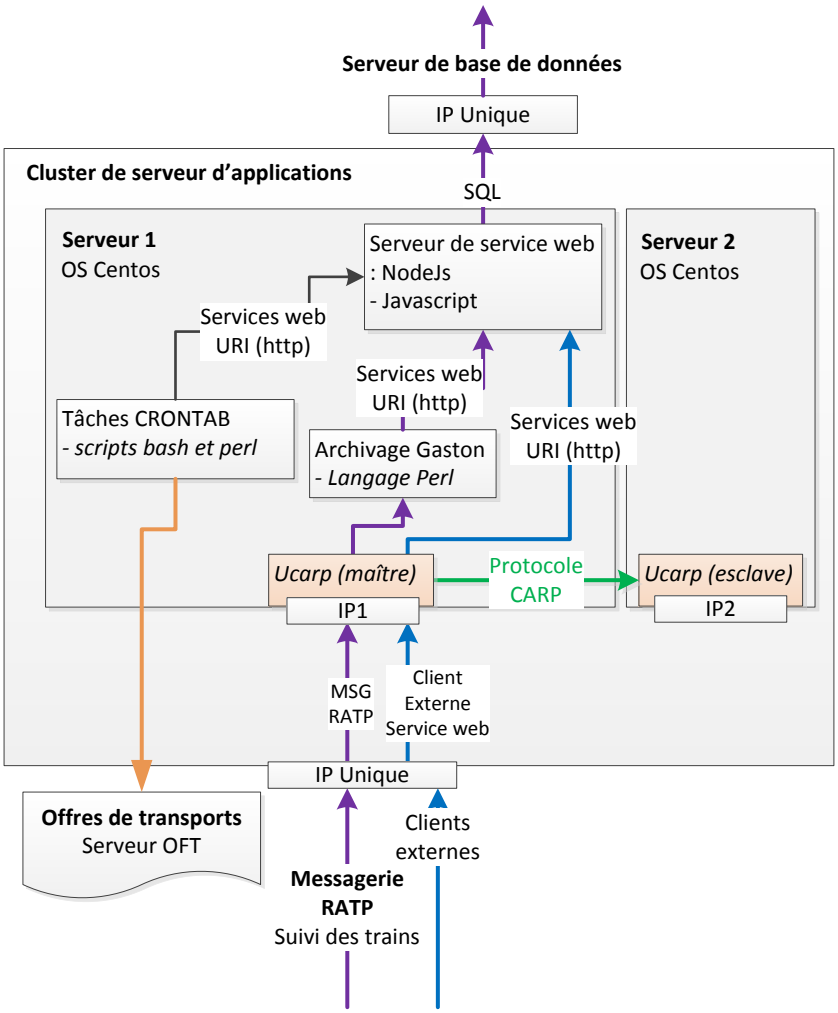


Figure 31 - Architecture logicielle du serveur d'application

Pour simplifier la présentation de l'architecture logicielle, je les ai séparées volontairement pour la présentation (*cf chapitre **4.4***).

4.3.3 Architecture matérielle de la gestion des applications

L'architecture du serveur d'application est moins complexe que le serveur de base de données du fait qu'il n'y a que la gestion de redondance à gérer.

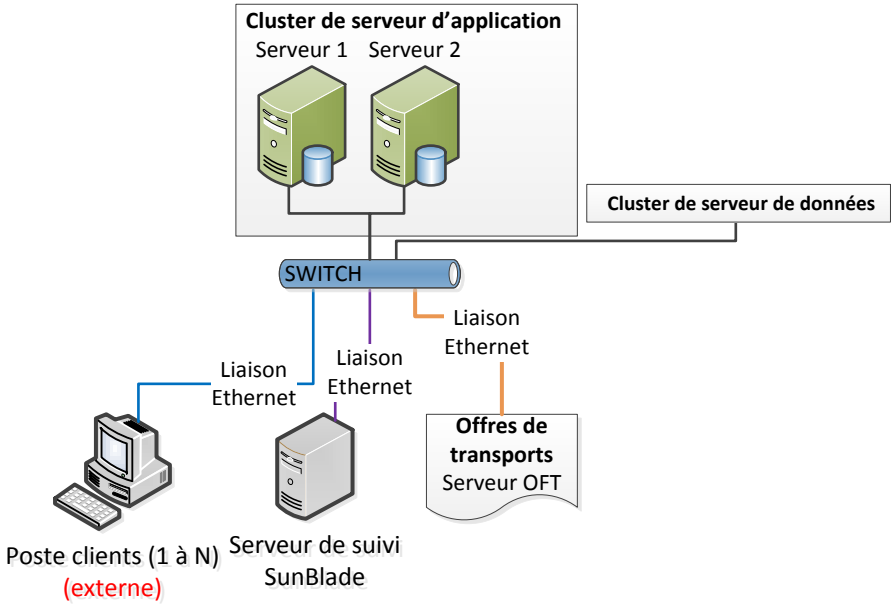


Figure 32 - Architecture matérielle du cluster des serveurs d'application

Pour des raisons de coûts et de préconisation, nous utiliserons des logiciels et des librairies venant du libre. Le serveur est installé avec un Linux CentOS. Nous uniformisons nos systèmes avec le même OS.

Pour simplifier la présentation de l'architecture logicielle, je les ai séparées volontairement pour la présentation (*cf chapitre **4.4***).

4.3.4 Processus d'archivage

4.3.4.1 Présentation

L'archivage de données récupère les informations du suivi des trains pour les insérer en base de données, il joue un rôle d'interface. Il se présente actuellement sous la forme de processus applicatifs.

Il existe actuellement deux processus installés pour la récupération des messages de suivi MSG99 et MSG100. Il y a message par processus ce qui permet la séparation des tâches. Ils sont échangés entre la base et le serveur de suivi des trains. Ce sont des processus écrits dans le langage C.

Pour la nouvelle structure, afin d'avoir un serveur de base de données uniquement dédié au stockage et à la fourniture d'information, les processus d'archivage sont transférés vers les serveurs d'applications.

Plusieurs problématiques s'imposent à nous.
La première consiste à n'impacter aucune modification sur les applications en connexion avec ce nouveau système. Ce qui veut dire que le serveur de suivi ne doit pas être impacté.

La seconde consiste à ne plus accéder directement à la base de données en exécutant des requêtes SQL mais en utilisant des services web.

Avec la volonté d'utiliser un système unique d'accès aux données et disponible de tous, l'archivage est un outil à utiliser temporairement. Il est prévu de faire la modification de chacune des applications ayant un besoin de données pour qu'elles utilisent les services web.

Pour simplifier mes développements mais aussi la maintenance de ce genre d'application, j'ai fait le choix de développer ce processus sous le langage PERL. Cela m'a permis de le rendre plus flexible en termes d'utilisation et de modification.

Je n'ai pas repris le processus existant qui est écrit en langage C. Cela vient du fait que pour exécuter les services web (URL/HTTP), il existe une librairie appelé « CUrl », c'est actuellement la plus connue. Cependant, les outils de compilation du processus d'archivage

sont anciens. La librairie CUrl est disponible uniquement pour des versions plus récentes. Ce qui signifie qu'il faudrait reprendre entièrement l'application.

Dans les deux cas (Perl ou C), il me fallait reprendre l'application pour utiliser les services web. Le langage Perl est plus adapté pour faciliter le traitement de chaîne de caractère mais c'est aussi un langage précompilé (compilation du code à l'exécution du processus) ce qui me permet de faire des modifications plus rapidement qu'une application écrite en C.

4.3.4.2 Conception d'un module d'archivage

Le but de ce processus n'est pas de révolutionner mais de simplifier la gestion du stockage et l'utilisation de services web. Nous sommes partis sur des solutions venant du libre.
Perl pour le développement du processus et CUrl pour l'appel des services web.

L'archivage actuel est un processus qui a été conçu en tant que serveur. Le client se trouve être le serveur de suivi des trains. On se retrouve face à un problème de conception, le processus qui possède la donnée est client.

Dans ce portage, on revoit ce principe entre le processus d'archivage et le serveur de suivi des trains en inversant la communication. Elle fonctionne par des messages TCP.

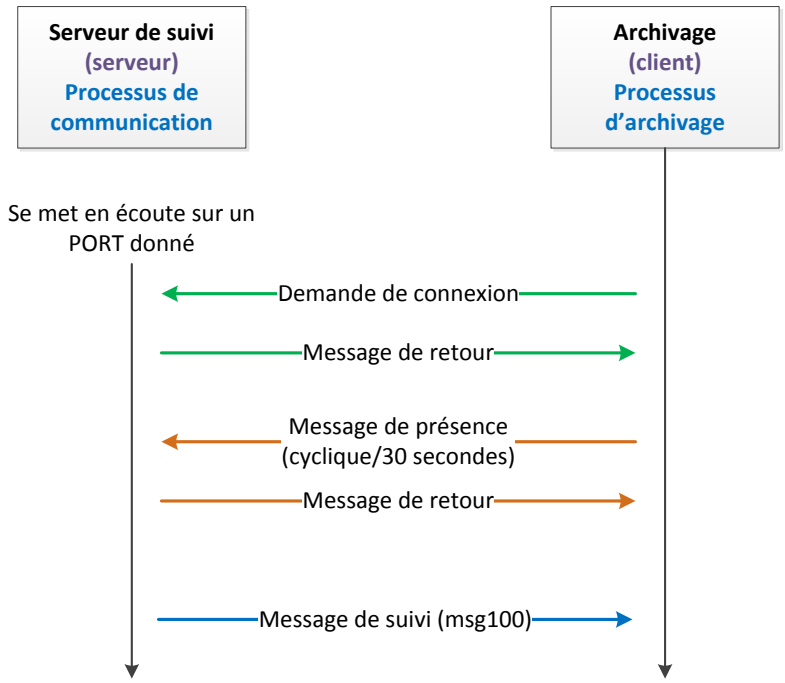


Figure 33 - Archivage : diagramme de séquence

Le serveur de suivi se met en écoute sur un port défini à l'avance dans son fichier de configuration. Le processus d'archivage (client) fait une demande de connexion sur ce port.

Lorsque le serveur l'accepte, il lui envoie les messages de suivi (MSG99 et MSG100) par la messagerie RATP.

L'inconvénient de ce système nécessite d'envoyer un message de présence au serveur de suivi toutes les 30 secondes.

Pour la réalisation du projet, je n'ai réalisé uniquement l'archivage du message 100. Il correspond au suivi des trains avec les occupations et libérations ainsi qu'aux arrivées et aux départs de chacune des circulations.

L'archivage porté reprend les fonctionnalités de l'archivage actuel avec de nouvelles fonctionnalités et des améliorations.

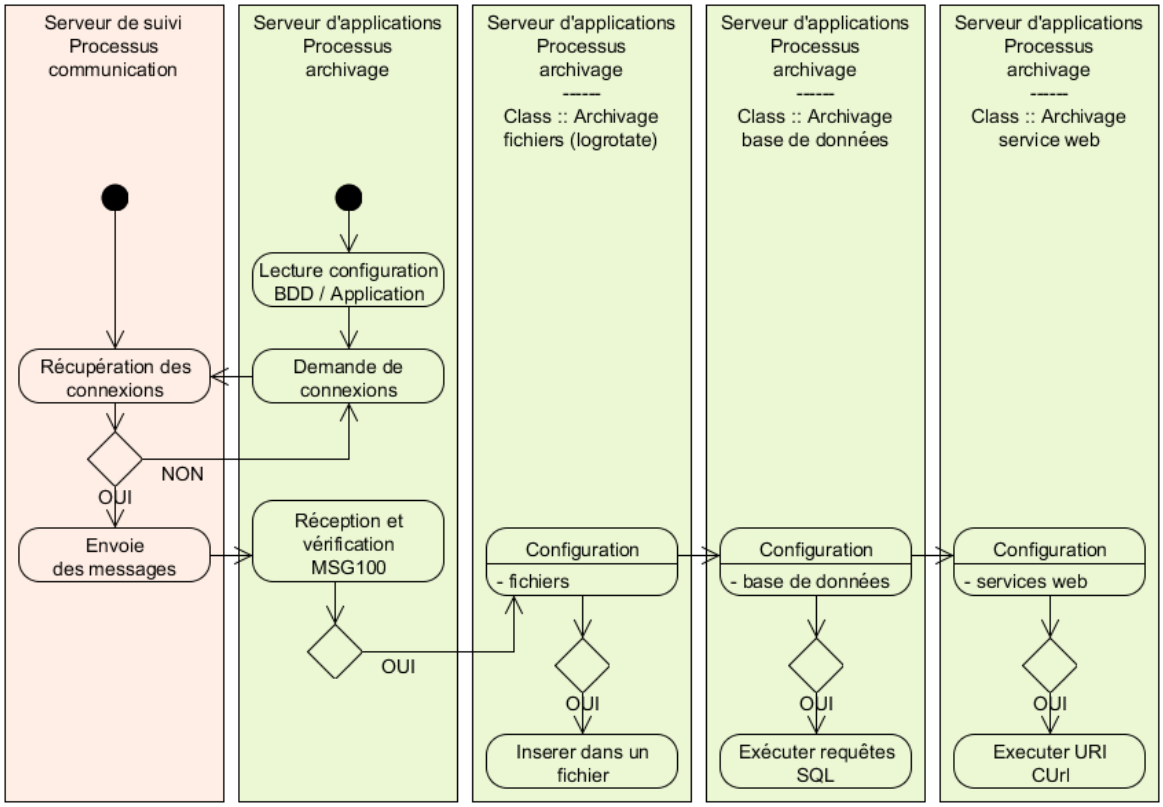


Figure 34 - Diagramme d'activité du processus d'archivage

La première fonctionnalité que je préserve correspond à l'archivage de la partie « Données » de la trame TCP utilisé dans la messagerie RATP. Je lui apporte une amélioration avec l'utilisation de la fonction « logrotate ».

Cette fonction existe par défaut sur tous les systèmes linux. C'est un utilitaire qui permet la gestion des fichiers de logs dans un système. Un certain nombre d'option sont implémentées telles que la rotation automatique et la compression des fichiers.

L'archivage des fichiers de logs est assuré selon un paramétrage du système pour éviter la saturation des disques.

Pour mon projet, dans une optique de gain de place, j'ai configuré la rotation automatique des fichiers selon des critères de journée. Je souhaite avoir le même nombre de jours d'archive en base et en fichiers. Je connais d'après la spécification qu'il faut stocker un an glissant de données. Après la réalisation d'un test d'archivage en fichier avec le processus actuel et celui porté, une journée d'exploitation en fichier possède une taille de 25Mo.

Avec cette solution, je m'assure de ne jamais saturer la taille du disque. En supplément, une option de compression est appliquée pour chaque fichier d'archive terminé, ce qui a pour but de réduire la taille des archives sur le disque.

La seconde fonctionnalité que je souhaite temporairement intégrer est de pouvoir insérer des données directement en base. Je l'ai portée pour effectuer des tests de comparaison.

La nouvelle fonction apportée correspond à l'utilisation de services web. Elle a pour but de remplacer les accès directement à la base de données. J'applique les mêmes règles d'accès pour nos applications et les applications externes. Cela uniformise les développements et les traitements.

4.3.4.3 Intégration du processus

De la même manière que la base de données, l'intégration du processus d'archivage se fait en parallèle de celui qui existe actuellement. La particularité c'est que l'on ne l'installe plus sur la base mais sur un serveur d'application. Une contrainte minime, il est nécessaire de modifier la configuration du serveur de suivi pour la prise en compte de l'inversion de communication entre l'archivage et le serveur de suivi. Cela nécessite une modification des fichiers de configuration du serveur et non du code dans lequel il faudrait recompiler l'application.

Les modules d'archivages permettent d'alimenter la base de données pour de multiples applications. La coupure des processus d'archivage existants ne se feront qu'une fois l'ensemble des systèmes portés vers la nouvelle architecture.

Les serveurs d'applications sont dans un cluster avec un maître et un esclave. La surveillance du processus est gérer par l'outil gérant la redondance des serveurs d'application UCarp (*cf chapitre **4.3.2***).

4.3.5 Services web

4.3.5.1 Présentation

J'ai fait le choix d'utiliser les services web pour la mise à disposition des informations dans l'ensemble de nos systèmes mais aussi de proposer une solution pour les insérer dans la base de données.

Ces services offrent une méthode simple et flexible de récupération des données pour d'autres clients externes de notre architecture.

4.3.5.2 Les technologies pour la conception de services web

4.3.5.2.1 Les langages de programmation

Les services web peuvent être réalisés de différentes méthodes et différent langages.
Le but est de proposer un développement simple et facilement évolutif.

Il y a une possibilité de développer des services web avec la librairie JAVA, n'ayant pas les connaissances nécessaires, il me semblait peu probable de fournir un système fiable dans sa finalité dans les délais de réalisation souhaité.

Ayant plutôt des connaissances dans les langages PHP et JavaScript, les deux solutions sont envisageables pour la réalisation de ces services web.

Le PHP liée à un serveur web de type Apache ou Nginx est une très bonne solution pour la gestion des URI. Il nous permet de les réaliser par un développement simple et rapide.

Le JavaScript pour sa part, nous le connaissons essentiellement du côté client exécuté sur un navigateur pour dynamiser nos pages web. Depuis quelques années, des librairies apparaissent et modifient son utilisation pour le placer côté serveur.
Une des plateformes les plus connues venant du libre se nomme Node.js. Elle se base sur le moteur d'exécution JavaScript V8[11] de Google qui permet d'exécuter très rapidement du code JavaScript.

[11] **Moteur d'exécution JavaScript V8** : Un site est dédié pour afficher les performances du moteur V8 - https://www.chromeexperiments.com/

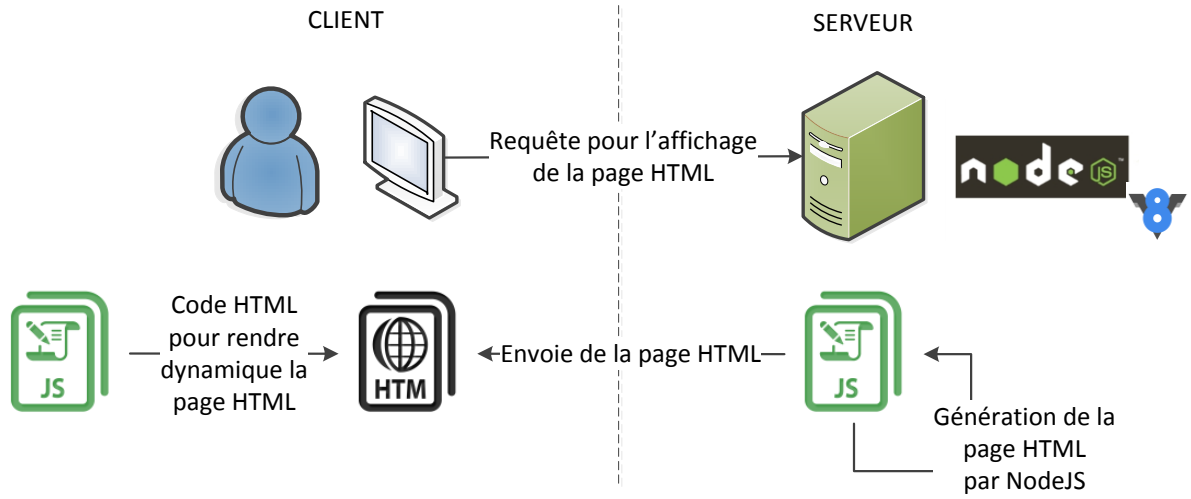


Figure 35 - Fonctionnement du serveur web Node.js

Dans son principe, Node.js ressemble à un serveur apache (PHP), cependant il offre une nouvelle approche en utilisant le langage JavaScript.

Node.js utilise le JavaScript sous forme événementiel non bloquante qui lui permet de réaliser des applications permettant d'exécuter plusieurs fonctions en parallèle contrairement à PHP. Ce qui permet d'améliorer fortement la montée en charge.

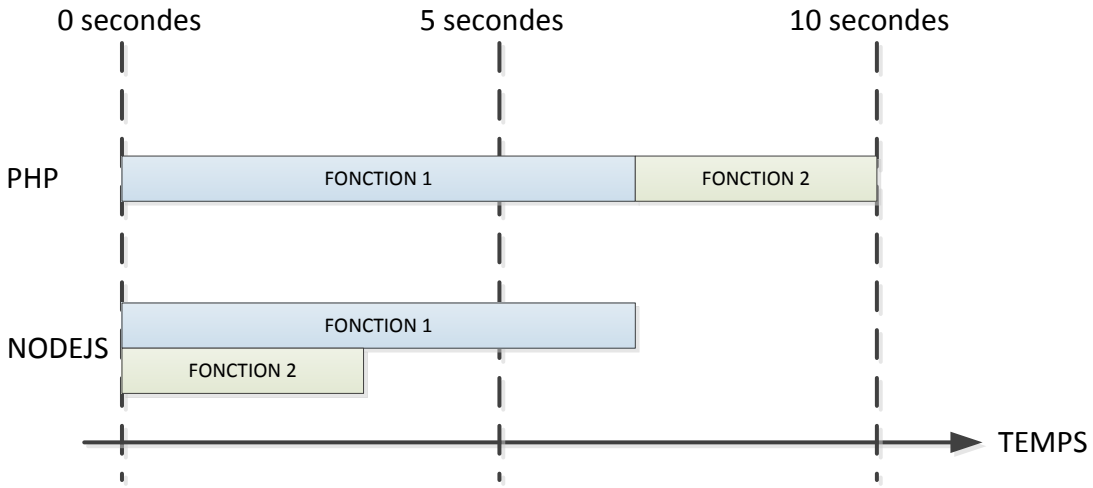


Figure 36 - Comparaison de fonctionnement entre Node.js et PHP

Contenant un module HTTP dans le même principe que Nginx, il permet se transformer en serveur web.

4.3.5.2.2 Le choix retenu

Bien que SOAP soit normalisé et plus mature. Dans notre cas, l'architecture REST est plus adaptée à notre situation de fourniture d'informations. Ce choix se porte aussi sur le fait qu'il permet une mise en place simple et rapide. REST propose une solution sans gestion d'état

ce qui nous enlève un traitement supplémentaire sur la sauvegarde d'informations en cas d'arrêt.

Pour la fourniture d'informations, nos services web ont la possibilité de les fournir en XML ou JSON. Pour éviter d'ajouter un argument dans l'URI, nos services web se chargeront de vérifier le « content type » pour la validation du type de format à utiliser. Pour mettre à disposition un système multi format, nous vérifierons le « content type » demandé par le client pour lui fournir l'un des deux formats possibles. Le content type est un attribut du protocole HTTP qui permet de définir le format d'échange de données.

Pour le langage de programmation, le choix retenu est la plate-forme Node.js pour la réalisation des services web.

Il y a deux raisons qui nous permettent de préférer du JavaScript par rapport à du PHP.

La première se porte sur les performances. Après plusieurs études de performance entre Node.js et des différents serveurs web couplés avec PHP, on s'aperçoit que Node.js est le plus performant. Cela est dû principalement à sa gestion événementielle mais aussi à une lecture ultra optimisée du JavaScript.

La seconde se porte sur le fait que l'on souhaite en parallèle fournir des applications de type temps réel pour l'affichage d'information. Cette méthode est possible si le serveur met à jour le client de façon événementiel sans rafraîchissement du navigateur ce qui est maintenant possible avec Node.js. Nginx ou Apache ne permettent pas ce genre de mise en place facilement.

4.3.5.3 Conception des services web

Les services web mis à disposition de chacun des clients pour accéder à la base de données concernent trois types d'applications.

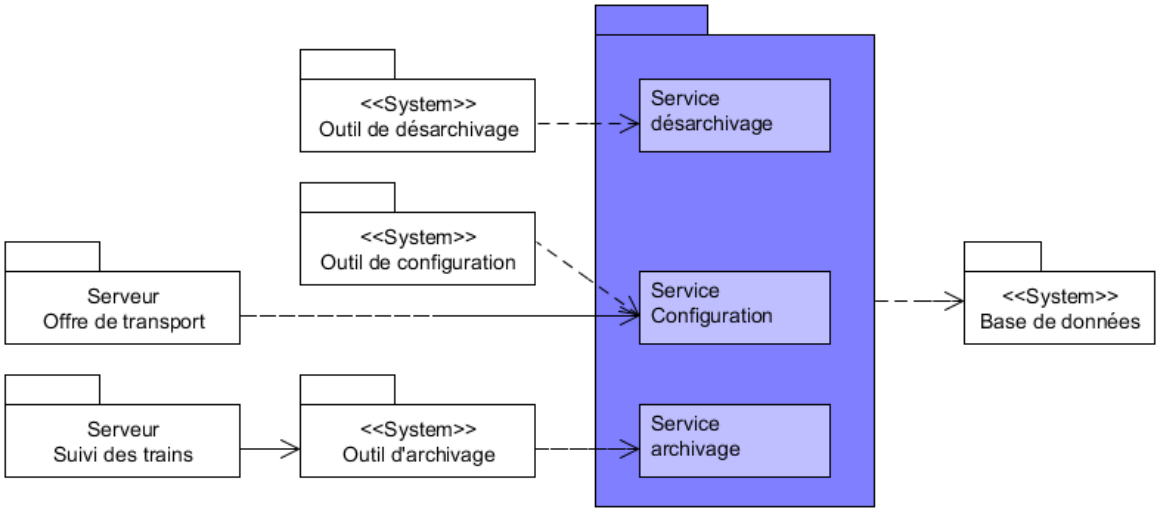


Figure 37 – Services web : Diagramme de package

Les packages sont décomposés sous formes de plusieurs URI.

Pour ne perdre aucune information dans la base de données, la fonctionnalité de suppression (DELETE) est verrouillée pour un certain type d'utilisateur (administrateur). Elle est utilisée pour des purges régulières dans la base de données. Seuls les trois autres seront disponibles : GET, PUT et POST.

Le package « **configuration** » contient les services web liés à la topologie et à l'offre de transport. Les clients accèdent aux données uniquement en lecture (GET), les autres fonctionnalités sont utilisées par le gestionnaire de données (IPL).

Le package « **archivage** » contient les services web liés à l'archivage. Les clients n'ont pas accès à ces services. La fonction d'insertion (POST) est utilisée par le processus d'archivage écrit en PERL et la fonction de suppression (DELETE) est utilisée pour la réalisation d'une purge.

Le package « **désarchivage** » contient les services web liés à la lecture des données archivés. Les clients n'ont accès aux données qu'en lecture (GET).

Pour la réalisation de ce projet, je me suis concentré sur la réalisation des packages « configuration » et « archivage ».

En priorité, j'ai défini les URI pour exécuter les services web.

Pour le package d'archivage, j'ai créé cinq URI permettant l'insertion des données du message de suivi (MSG100). Pour rappel, ce message contient cinq évènements. Ces URI sont utilisées par le processus d'archivage avec la fonction POST (insertion) :

- Evènement « Occupations » : http://127.0.0.1:8082/archivages/occupations
- Evènement « Libérations » : http://127.0.0.1:8082/archivages/liberations
- Evènement « Arrivées » : http://127.0.0.1:8082/archivages/arrivees
- Evènement « Départs » : http://127.0.0.1:8082/archivages/departs
- Evènement « Horaires » : http://127.0.0.1:8082/archivages/horaires

Pour le package de configuration, j'ai créé deux URI permettant l'accès aux données. Une accède aux données des graphiques horaires (données théoriques) pour effectuer une comparaison avec les données temps réel et l'autre accède aux données des gares. Les accès ce font avec la fonction GET (lecture) :

- Graphiques horaires : http://127.0.0.1:8082/offrestransports/graphiquehoraires
- Gares : http://127.0.0.1:8082/topologies/gares

Node.js est une plate-forme sur laquelle il faut développer l'ensemble du serveur et de l'application. Pour simplifier sa conception, j'ai décomposé l'application en plusieurs parties/fonctions qui se présente dans l'application sous forme de répertoire.
Je les ai nommées :

- Configuration
- Modules
- Routes
- Tests

Le répertoire « **configuration** » contient les données d'accès à la base de données et les données liées à l'application.

Le répertoire « **modules** » contient les deux fonctionnalités essentielles pour les services web. La première fonctionnalité correspond aux fonctions d'accès à la base et la seconde correspond à l'envoie des données suivant le format voulu.

Le répertoire « **Routes** » contient l'ensemble des URIs exécutable par les différents clients.

Le répertoire « **Tests** » contient l'ensemble des tests unitaires appliqués sur les URIs pour les valider.

Le fonctionnement d'un serveur Node.js est relativement simple. Lorsque le client souhaite accéder aux données, il exécute une URI depuis un navigateur ou par l'application CUrl.

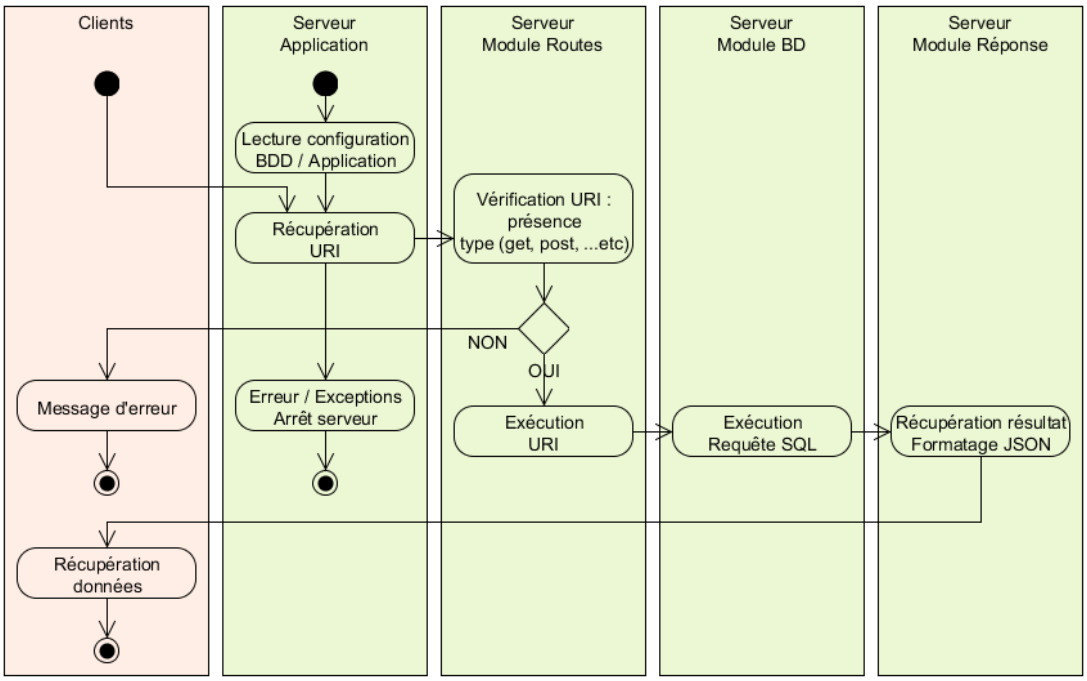


Figure 38 - Diagramme d'activité du serveur Node.js

Après avoir récupéré l'URI, Il faut vérifier sa présence par le gestionnaire de route. Si elle n'existe pas, l'application envoie un message d'erreur au client. Si cette URI existe, le gestionnaire de route exécute la fonction qui lui est associée.

Dans mon cas, l'ensemble des URI accèdent à la base de données. J'ai créé des fonctions liées pour chaque URI dans le cas où elle serait appelée en GET ou en POST.

Lorsque je récupère le résultat, je le retourne au client via une fonction de formatage (ex : JSON) nommée simplement « reponse »

4.3.5.4 Intégration des services web

Les services web n'ont pas d'équivalence dans la base de données actuelle. Ils sont installés sur les serveurs d'application de la nouvelle architecture.

Les services web sont gérés par le serveur Node.js. Il est prévu d'installer un système de surveillance permettant en cas de panne du processus Node.js de passer la main à un autre serveur d'application. Les serveurs étant en cluster, le maître laisserait sa place à un des serveurs secondaires.

4.3.6 GASTON : Application de graphiquage

4.3.6.1 Présentation

Cette application a pour objectif de fournir un outil d'enquête et d'investigation sur les causes de retard des trains RER. C'est une application qui actuellement est sous forme de client lourd. L'objectif est de porter cette application sous forme de client léger temps réel.

Elle affiche principalement un graphe de suivi d'exploitation, dénommé graphicage automatique, dans un repère espace-temps.
D'autres fonctionnalités dites de « Qualité Trafic » présentent les écarts, les intervalles et les temps de stationnement en relation avec le graphiquage affiché.

Ces fonctions peuvent être visualisées avec deux modes différents :

- **Le mode instantané** : permet de visualiser les données de suivi de la journée en cours, avec rafraîchissement au cours du temps.
- **Le mode différé** : De la même façon permet d'afficher non pas au cours du temps mais sur les journées sauvegardées en base.

Les données permettant d'alimenter l'application sont de trois types :

- Les données réelles, qui serviront à construire les courses de **l'horaire réel.**
- Les données trains, qui serviront à construire les courses de **l'horaire théorique**.
- Les données dessertes, qui serviront à remplacer les données trains si nécessaire.

L'application présentée dans ce mémoire ne se charge que de la partie : **mode instantané**. La partie différée sera réalisée ultérieurement.

J’ai choisie de présenter dans ce mémoire la partie temps réel de l’application web. Le temps réel pour une application web représente un challenge personnel mais aussi technique. Les technologies web d’aujourd’hui permettent la réalisation d’applications ultra-rapides qui vont me permettre ainsi que pour notre entité de modifier et faciliter l’accès aux données.

4.3.6.2 Conception de l'application web

Cette application possède deux modes, il y a le temps réel (instantané) et le temps différé. La différence entre les deux modes provient de la gestion de récupération immédiate des données de suivi ou de la récupération des données suivants une journée souhaitée.

Les autres fonctionnalités sont similaires.

Pour chaque journée, il est important de pouvoir comparer les données réelles d'une journée avec les données théoriques obtenues dans le graphique horaire appliqué pour la journée souhaité. L'exploitant à la possibilité de modifier l'offre de transport en cours de journée en cas d'anomalie ou de dysfonctionnement sur la ligne. Il faut donc recharger les données théoriques d'une journée suivant le graphique horaire choisi.

Pour la réalisation de cette application, il faut choisir une technologie adaptée pour ce genre de système. Lorsque l'on réalise une application web, on se dirige dans la plupart des cas vers le PHP. Dans mon cas, je préfère reprendre Node.js qui est utilisé pour les services web. Cela minimise le nombre de technologies employées dans le système. Node.js permet aussi de réaliser des applications clientes temps réel.

Pour cela, j'utilise une librairie appelée « socket.io » sous Node.js, elle va permettre d'ouvrir une connexion entre le client et le serveur.

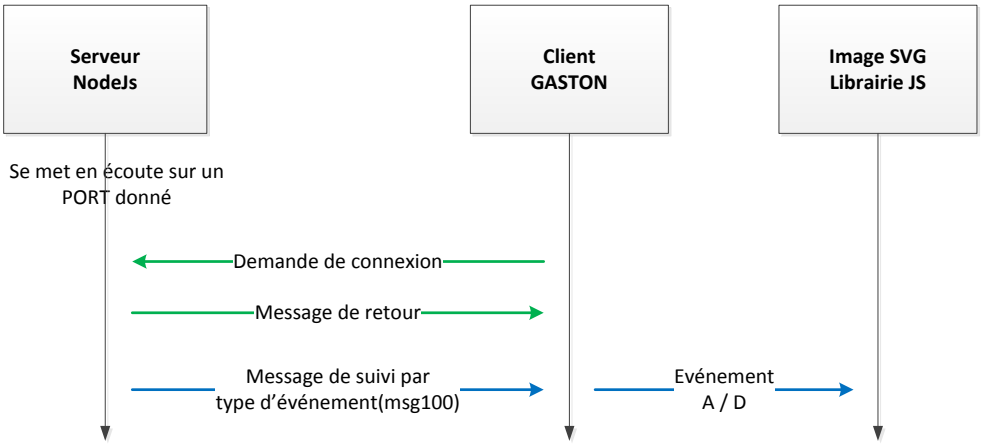


Figure 39 - Diagramme de séquence du traitement des informations de l'application Web Gaston

Les événements d'arrivée et de départ sont envoyés à l'application cliente sans rafraîchissement du client web. Le client web s'assure de l'événement à traiter et l'affiche dynamiquement sur l'image SVG par l'intermédiaire d'une librairie graphique.

Pour l'affichage des pages HTML, j'utilise un moteur de template qui se nomme EJS. Ces moteurs de template permettent de simplifier le développement des pages web pour séparer la partie graphique du développement. Ces moteurs sont très utilisés dans le développement de site web. Dans notre entité, nous utilisons tous types de langages pour nos projets, nous n'avons jamais la possibilité d'avoir une expertise sur un domaine. Il important de proposer des méthodes simples mais efficaces pour nos applications.

L'application GASTON affiche avant tous des graphiques sur le suivi des trains. Avec le web, j'avais plusieurs choix pour les réaliser. Il y a le format SVG permettant la réalisation d'image vectorielle et le format CANVAS récemment implanté comme composant du HTML dans la spécification du HTML5 validé en 2014 pour la réalisation de bitmap.

Les avantages du SVG se basent sur XML pour son format de données. Il permet aussi la création d'image vectorielle. Il a pour particularité d'être standardisé par le W3C depuis 2001 alors que le Canvas ne l'est que depuis 2014.

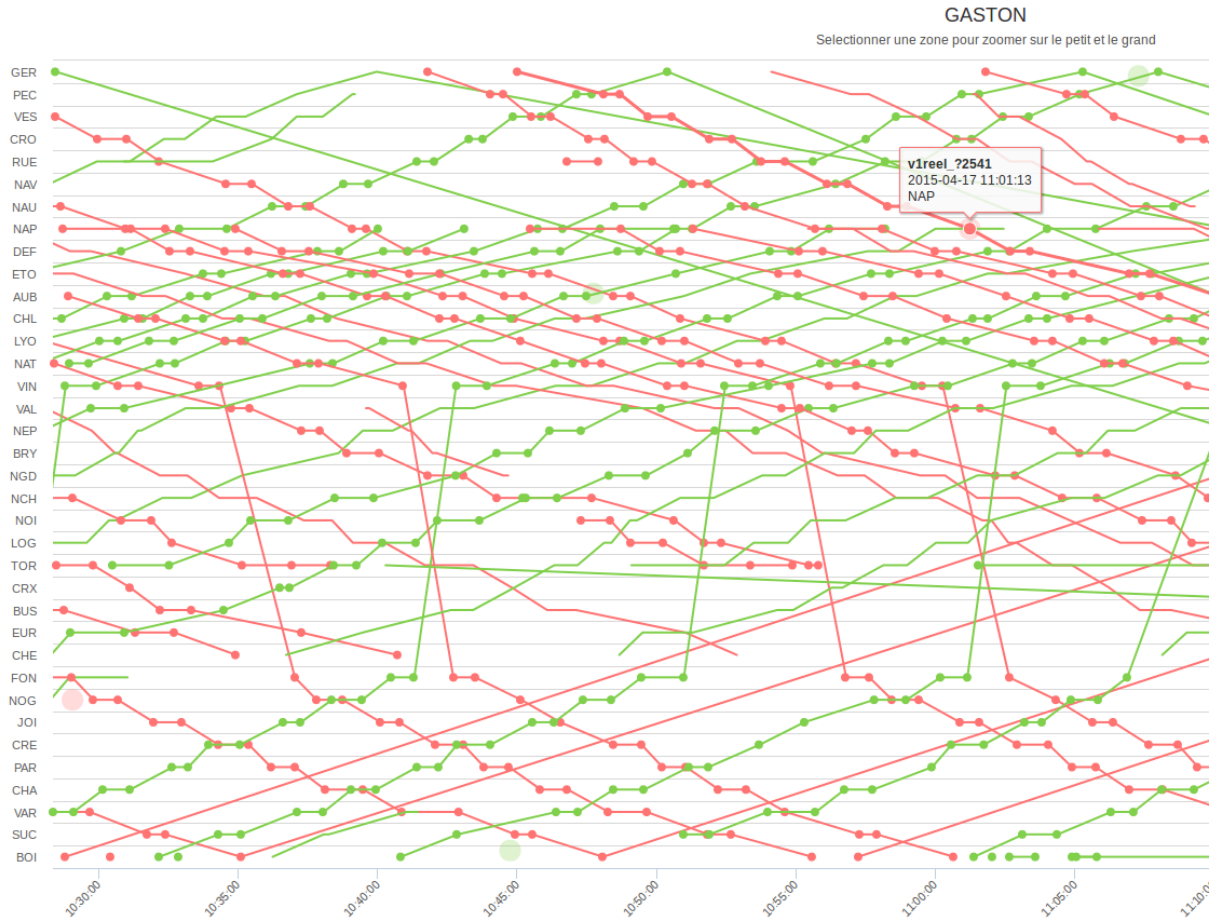


Figure 40 - Exemple d'un graphique obtenu via l'application en temps réel.

On retrouve dans l'axe dans l'axe des abscisses le temps et dans l'axe des ordonnées les gares. Les courbes rouges correspondent au suivi des missions en voie 1 et en vert, les missions en voie 2.
C'est une application qui permet d'un coup d'œil de vérifier le cheminement complet de chacune des missions pendant une journée.

4.3.6.3 Intégration de l'application web

L'application web développée avec Node.js sera installée sur les serveurs d'application.
Utilisant les données de l'archivage de suivi des trains avec le MSG100.

Elle n'impacte pas l'application existante. Elle peut être installée en parallèle de l'existante. Cependant, il est nécessaire d'avoir mis en place le processus d'archivage et les services web pour faire fonctionner l'application. Elle est dépendante de certaines données contenues dans la base de données.

4.4 Synthèse de l'architecture matérielle et logicielle

L'architecture cible réorganise l'échange des données pour simplifier la récupération et l'accès aux données.

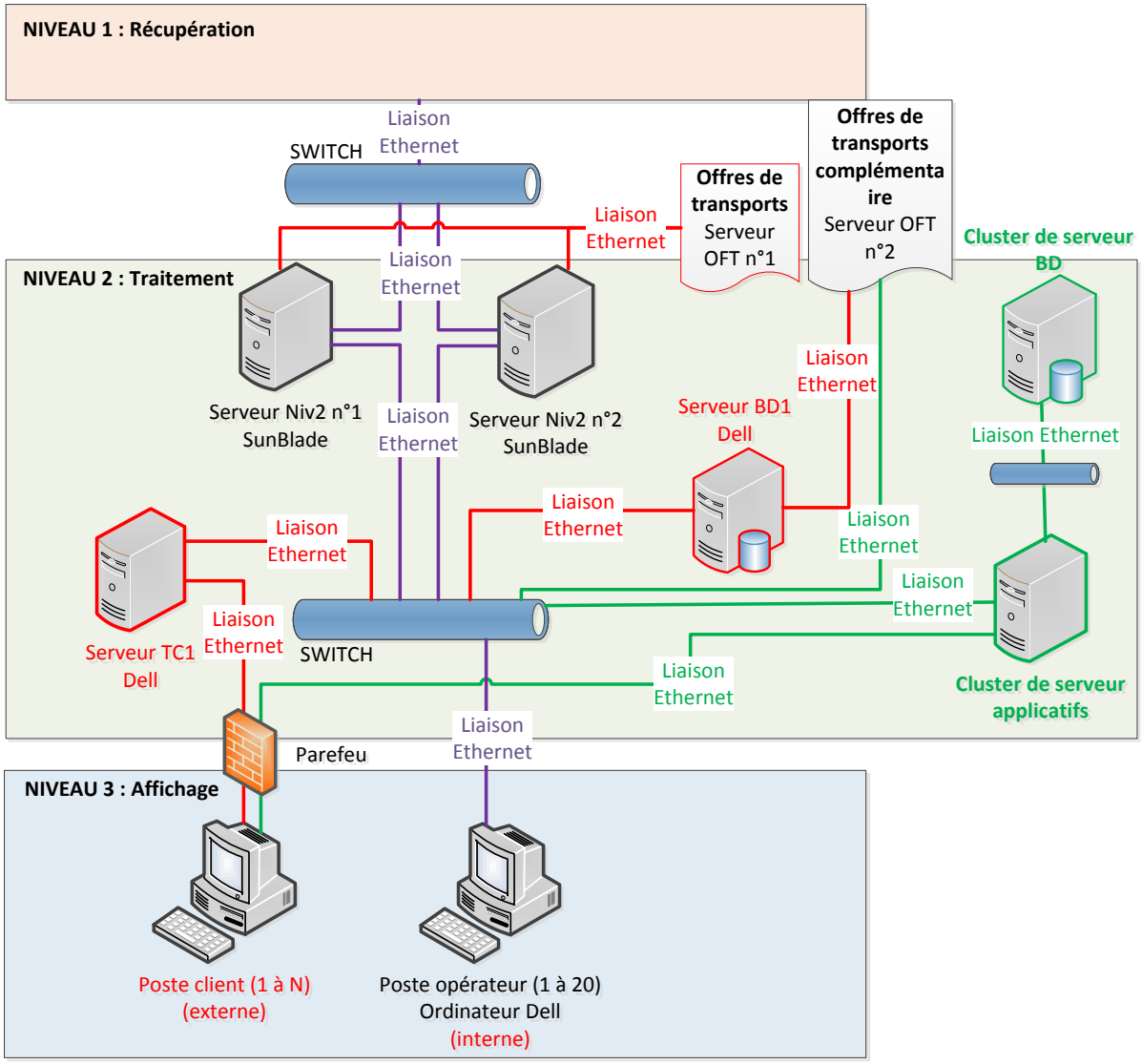


Figure 41 - Architecture matérielle de la cible

En comparaison avec l'architecture existante, je supprime le serveur TC1 ainsi que l'accès à l'offre de transport utilisé par le serveur de suivi. De même la base de données est isolée, elle devient accessible uniquement par un serveur d'application.

L'offre de transport du serveur de suivi ainsi que les données pour les clients externes sont fournies par le serveur d'application.

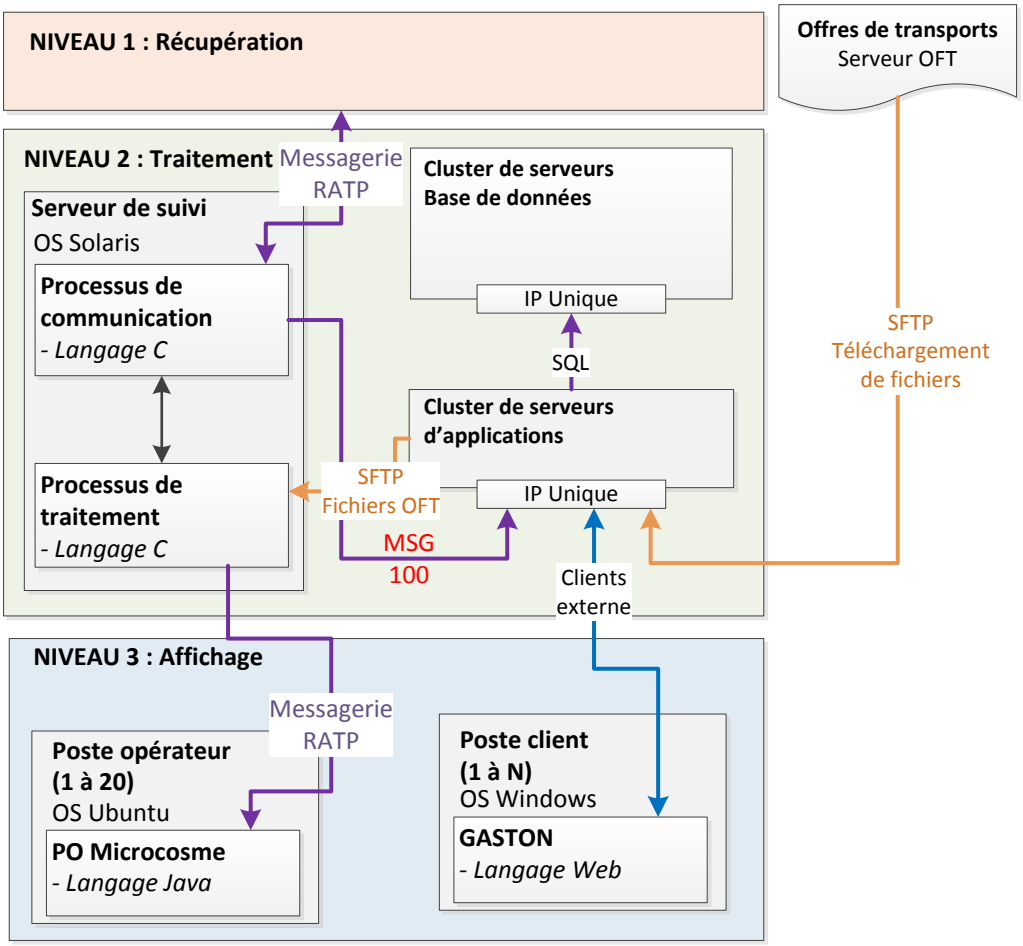


Figure 42 - Architecture logicielle de la cible

L'unique modification en dehors du serveur d'application et de la base de données vient des postes clients externes. Il y a la suppression des applications Java pour des applications web. De plus, l'échange des données se fait par des services web et non plus par la messagerie RATP.

Quelques changements affectent le fonctionnement existant de l'architecture. Le plus important vient de l'utilisation des services web pour l'accès aux données contenu dans la base de données.

L'offre de transport est mise à disposition pour le serveur de suivi automatiquement par des scripts exécutés sur le serveur d'application.

Le dernier changement vient des clients externes qui ne se connectent plus sur TC1 mais sur le serveur d'application pour obtenir les données contenues en base.

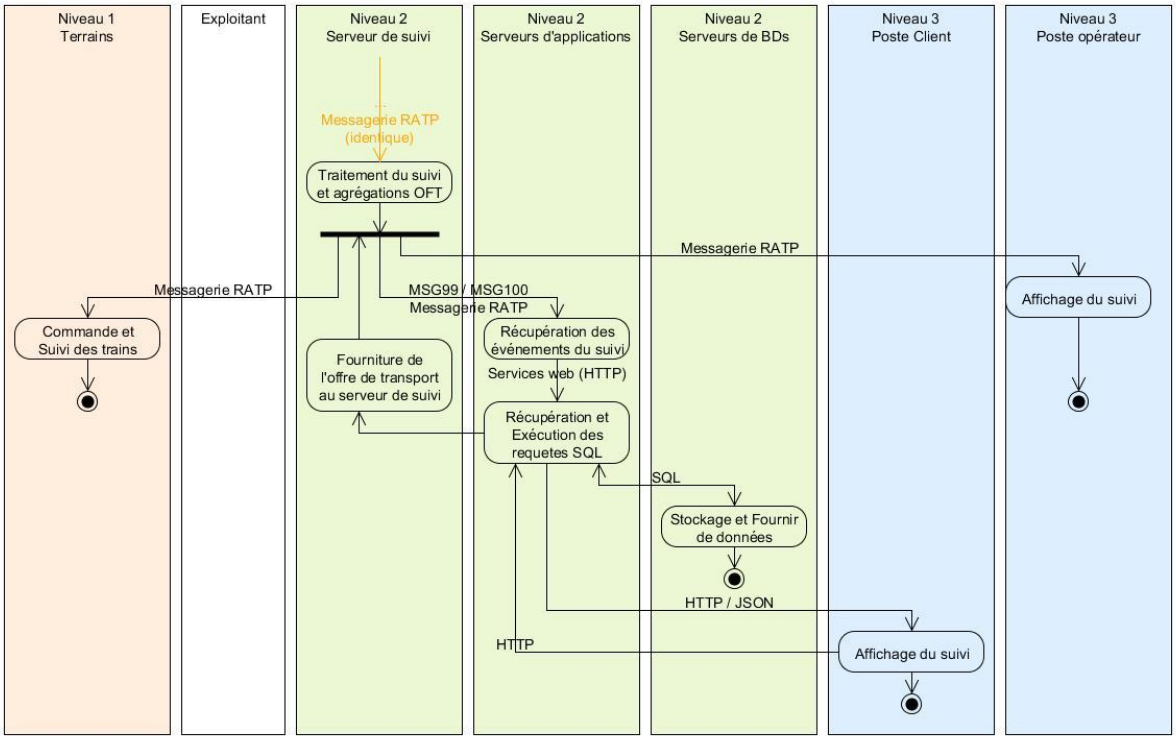


Figure 43 - Diagramme d'activité de l'architecture logicielle cible

4.5 Le plan de test

4.5.1 La méthodologie

La volonté de ce projet est de présenter un ensemble d'applications permettant de réaliser une chaîne complète du suivi des trains, de la récupération à l'affichage.

Dans notre service de maintenance, nous avons souvent été amenés à réaliser toutes les phases d'un projet. Par expérience, on constate que la phase de test est généralement bâclée par un manque de temps.

Les tests ont pour but de réaliser un produit avec zéro défaut. C'est un processus qui doit représenter normalement un peu moins de la moitié du temps d'un projet. Ils deviennent un atout pour minimiser les interventions de maintenance logicielle qui deviennent nombreuses avec le temps.

Pour ce système qui a pour objectif d'être ouvert à tous, il est important de prendre le temps nécessaire pour les appliquer. Il existe 4 applications à tester. Il y a la base de données, l'archivage, les services web et l'application GASTON. Les services web ont nécessité une plus grande attention du fait de l'importance de cette fonctionnalité dans ce système.

Pour l'installation en production de mon système, je dois exécuter plusieurs familles tests :

- Les tests unitaires
- Les tests fonctionnels
- Les tests de non-régression
- Les tests de performances

Pour des raisons de temps, l'ensemble des tests non pas été finalisés et les tests de performance n'ont pas été traités dans ce projet.

J'utilise mon poste de développement pour la réalisation des tests, je n'ai pas reçu les serveurs pour le moment. C'est un ordinateur portable HP 6470. Il possède 4Go de mémoire et un processeur Intel® Core™ i3-3110M CPU @ 2.40GHz × 4. Le disque dur a une taille de 113Go.

Pour tester unitairement mes applications. J'utilise des modules à installer dans chaque application. Ces librairies sont différentes suivant le langage. Pour l'application d'archivage écrite en langage Perl, j'utilise une librairie qui se nomme « **Test::More** ». Pour les services web, j'utilise la librairie « **vows** ». Cette librairie permet des tests d'insertion ou de récupération en base de données.

Pour les tests fonctionnels, j'utilise un outil web qui se nomme « **TestLink** ». Elle a pour avantage d'être open source et d'être utilisable en tant que client léger (web). C'est un outil simple et rapidement utilisable. Elle a pour but de simplifier la création des fiches de test écrites dans document Word. Cet outil n'automatise pas les tests.
Elle possède plusieurs avantages :

- Les fiches de test sont disponibles par toute l'équipe via l'intranet.
- Les fiches peuvent être rejouées indéfiniment avec un historique archivé.
- Assignation des tests aux différents utilisateurs
- Une génération automatique d'un document des tests

Suite à son utilisation sur mon projet, c'est un outil qui est maintenant utilisé pour l'ensemble de nos projets dans notre entité.

Pour les tests de non régressions, je les ai intégrées aux tests fonctionnels pour valider que les deux outils que j'ai portés fournissent le même résultat que les applications d'origine.

Je détaille dans les chapitres suivants les tests unitaires et fonctionnels des applications du serveur de base de données et du serveur d'applications.
Ils ont été réalisés avec un jeu de données composées de différents types d'archives du MSG100 (occupations, libérations, arrivées, départs et horaires).

4.5.2 Les tests de la gestion des données

4.5.2.1 Les tests de la base de données

Les tests concluants :

- Installation de la base de données
- Insertion des droits d'accès à la base de données
- Insertion des données temporaires pour le nouveau modèle de données
- Reconstruction des données topologiques
- Insertion des données de l'offre de transport

Les tests non concluants ou non réalisés :

- Prise en compte de la réinsertion des données. Je supprime l'ensemble des données pour le moment pour éviter les doublons dans les données topologiques. Pour les données de l'offre de transport, je contrôle la présence du nom du graphique horaire, s'il est présent, je ne le réinsère pas.

4.5.2.2 Les tests de l'application PGPool II

C'est une application que j'ai installée sur deux machines virtuelles me permettant de simuler 2 serveurs de base de données.

Les tests concluants :

- Installation de l'application PGPool II en liaison avec PostgreSQL
- Installation d'un serveur Maître et d'un serveur Esclave
- Redémarrage de l'application en cas de relance d'un serveur
- Fonctionnement inchangé en cas de coupure du serveur esclave

Les tests non concluants ou non réalisé :

- Donner la main au serveur esclave en cas de coupure du serveur maître
- Vérifier de la répartition de charge
- Vérifier de la réplication des données

4.5.3 Les tests de la gestion des applications

4.5.3.1 Les tests du processus d'archivage

Les tests concluants :

- Chargement de la configuration
- Tentative de connexion au serveur de suivi toutes les 20 secondes en cas d'absence.
- Connexion TCP avec le serveur de suivi
- Envoi d'un message de présence (signe de vie) au serveur de suivi
- Récupération des messages RATP (MSG100) uniquement du serveur de suivi en exploitation
- Insertion des messages dans un fichier binaire
- Insertion des messages directement dans la base de données
- Exécution des services web pour l'insertion des données en base de données
- Non possibilité d'exécuter 2 processus d'archivage en même temps

Les tests non concluants :

- Connexion à deux serveurs de suivi. Un des serveurs est en exploitation et l'autre en redondance.
- Prise en compte de la redondance des serveurs de suivi.

4.5.3.2 Les tests des services web

Les tables de la base de données sont maintenant en relation entre elles. Il est plus délicat de tester objet par objet sans en impacter un autre. Cela est dû aux clés étrangères du moteur PostgreSQL qui fait un contrôle d'intégrité. Les tests ont pu être effectués selon des jeux de tests suivant un ordre hiérarchique précis.

Les tests unitaires et fonctionnels réalisés porte sur l'exécution des services web liés à l'archivage permettant l'insertion des données dans la base.

Les tests concluants :

- Exécution des 5 services permettant l'archivage des 5 événements associés au MSG100
- Envoi des données d'archivage temps instantané aux clients web GASTON connecté.
- Exécution du service permettant la récupération des gares dans la topologie

Les tests non concluants (à finalisé):

- Exécution des services web de l'offre de transport
- Exécution des services web de la topologie

4.5.3.3 Les tests de l'application GASTON

Les tests concluants :

- Connexion avec le serveur Node.js par une socket
- Récupération des données instantanées via la connexion socket entre le serveur et le client Node.js
- Affichage les données temps réalisés avec les données temps théoriques en mode différé.

Les tests non concluants ou non réalisés :

- Afficher les données « temps réalisé » avec les données temps théoriques en mode instantané. La librairie graphique utilisée est extrêmement gourmande, cela est due au rafraîchissement à chaque récupération d'un événement.
- Prise en compte de l'événement d'un nouveau graphique horaire en mode temps instantané.

4.5.3.4 Les tests de l'application UCarp

C'est une application qui n'a pas été installée pour le moment.
Les tests attendus seront réalisés avec deux serveurs dans le cluster.

Les tests concluants :

- néant

Les tests non concluants ou non réalisés :

- Installation de l'application UCarp
- Redémarrage de l'application en cas de relance d'un serveur
- Mise en place d'un serveur Maître et d'un serveur esclave
- Donner la main au serveur esclave en cas de coupure du serveur maître
- Fonctionnement inchangé en cas de coupure du serveur esclave
- Aucun flux vers le serveur esclave

4.6 Le déploiement

Il consiste à installer la nouvelle architecture sur la plate-forme de production. Cette opération est prévue fin d'année 2015. L'ensemble des applications en liaison avec la base actuelle n'ont pas été portées du fait qu'elles n'étaient pas traitées dans ce projet de mémoire. La migration se fera donc par étape.

La première étape consiste à mettre en parallèle les nouveaux serveurs de base de données et d'applications (*matériel* + *logiciel*) avec l'existant. Une modification de configuration sur le serveur de suivi est nécessaire pour l'envoi des messages vers le nouveau processus d'archivage. Cette solution me permet de garder intacte la chaîne existante afin qu'elle continue à fournir les informations nécessaires aux différents services. Il existera une chaîne principale (architecture existante) et une chaîne secondaire me permettant de valider le fonctionnement mais aussi la robustesse de la nouvelle architecture aussi bien sur les outils réalisés mais aussi les matériels installés.

Une seconde étape consiste à porter, tester et installer les autres applications non traitées.

L'arrêt de la chaîne principale se fera lorsque l'ensemble des fonctionnalités seront portées et mises en place sur la nouvelle architecture. L'arrêt est définitif après validation de la maintenance et de la maîtrise d'ouvrage.

Cette méthode permet dans le cas d'un dysfonctionnement de ne pas impacter l'existant.

4.7 La maintenance

Les dysfonctionnements peuvent occasionner des pannes matérielles ou applicatives. Sur ce projet, une maintenance est mise en place pour appliquer un ensemble d'actions afin de remettre le système dans son état nominal.

La nouvelle architecture ne modifie pas les applications de surveillance et surtout nos méthodes de maintenance. Cependant, il nous faut appliquer de nouvelles règles pour l'administration de la base de données mais aussi des outils de redondance logicielle.

La première action mise en place est préventive. Elle correspond à la surveillance de l'état de chaque processus applicatif et de l'état de chaque équipement (*disques durs, cpu, …etc*). Cette surveillance est faite par l'intermédiaire d'une application installée sur un serveur externe, l'application s'appelle Nagios. Elle utilise le protocole Simple Network Management Protocol (SNMP).

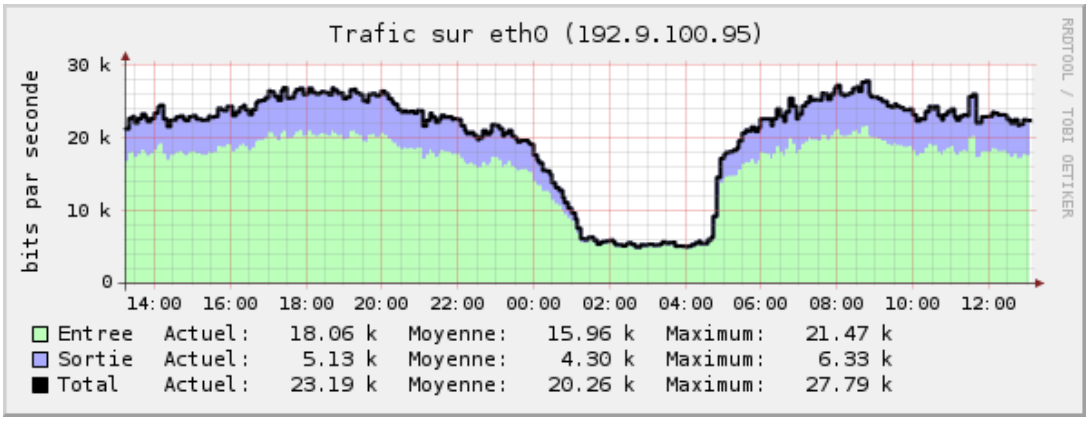


Figure 44 - Exemple du trafic sur la base de données existante par l'intermédiaire de l'outil Nagios

C'est un exemple de la représentation de l'utilisation CPU de la base existante. On s'aperçoit que la charge CPU est quasi nulle entre 1h30 et 4h30. Cette période correspond à la fin et début de service voyageur.

La seconde action est de relancerun système en cas de panne. Cette étape est réalisée par l'entité MSR-SAE. Si la panne est matérielle, ils ont à leur disposition des machines de réserve pour le remplacement. Si la panne est applicative, nous mettons à disposition des scripts de relance simplifiée, pour faciliter sa remise en marche. Cependant, il se peut qu'une relance ne soit pas possible. Nous leur fournissons à chaque version un livrable complet de

l’application qu’ils peuvent réinstaller. Un manuel d'installation leur est fourni pour suivre une procédure étape par étape.

La troisième action permet de remettre en place de façon permanente un système. Après avoir remi provisoirement en fonctionnement un système, il est important de comprendre et de corriger la panne. Pour nos recherches, j’ai conçu pour chaque application un système de log. Il enregistre l’ensemble des informations nominales (accès) et critiques (erreurs). Il nous permet de détecter, dans la plupart des cas, les événements qui pourraient avoir déclenché le dysfonctionnement. Dans le cas inverse, nous disposons d'une plate-forme de développement sur laquelle nous rejouons la journée d'exploitation pour essayer d'en localiser la panne (processus de rejeu).

La dernière action consiste à réaliser des sauvegardes journalières de la base de données vers un système de stockage extérieur. Cela permet en cas de crash total de pouvoir récupérer l'ensemble de la base de données à J-1.

4.8 Les avantages et contraintes

L'architecture mise en place permet de corriger les contraintes énoncées précédemment. Elles se concentrent essentiellement sur la gestion des données dans l'ensemble de notre architecture.

L'un des points forts de ce projet est de permettre la centralisation et l'unicité de l'ensemble de nos données en un point unique.

L'accès à ces données est crucial pour le fonctionnement global des applications. Avec deux systèmes en cluster (*serveurs de base de données, serveurs d'application*), nous assurons une haute disponibilité avec ce mécanisme. La redondance matérielle étant invisible du client. De plus, une gestion de répartition de charge logicielle nous permet d'être plus performants en termes de temps de réponse sur la base de données

Nous facilitons la fourniture d'informations avec la mise en place d'un système de services web. La méthode REST appliquée pour ces services rend interopérable notre système avec les autres clients. Nous utilisons des standards normalisés pour l'échange des données pour que chacune des applications puissent les récupérer et les traiter sans contraintes.

Les applications d'affichage de type client lourd sont remplacées par des clients légers (*web*). La maintenance se simplifie avec la suppression d'une gestion cliente et de mises à jour de données et graphique laborieuses.

Ils existent cependant certaines contraintes. La redondance des serveurs est logicielle. Il est nécessaire d'effectuer une surveillance plus rigoureuse de ce mécanisme.

L'accès à la base de données devient strict. Il n'est plus possible de réaliser directement des manipulations sur les serveurs. Cela vient du fait que le moteur PostgreSQL est sous la contrainte d'un réplicateur de données. Il est sensible à la moindre différence qu'il pourrait localiser entre les deux bases.

5 Bilan du projet

5.1 Apport personnel et difficultés rencontrées

Dans ce projet, l'apport personnel s'appuie sur mes connaissances en base de données et sur le développement d'applications web. Elles m'ont permis de proposer et de mettre en place les bases de l'ensemble de l'architecture.

Ces aptitudes sont appliquées dans mon quotidien à la RATP. Les différents projets auxquels j'ai pu participer, m'ont permis d'être plus prudent et plus méthodique.

Un projet de cette envergure n'est pas courant dans notre service de maintenance. Il a fait l'objet de nombreux échanges entre collègues. Ils ont des visions parfois très différentes, ce qui m'a permis d'en exploiter certaines idées intéressantes pour la réalisation de l'architecture.

J'ai utilisée des technologies et des méthodes intéressantes qui permettent d'envisager l'amélioration et l'évolution d'autres projets.

Malgré tout, j'ai eu avec quelques difficultés pour la confection de ce projet.

La première difficulté a été de réaliser un système n'impactant aucune application en relation avec la base de données actuelle. Ce qui signifie que des palliatifs temporaires étaient nécessaires dans beaucoup de cas.

La seconde difficulté correspond au manque temps. C'est un projet conséquent qui a demandé beaucoup d'attention et de préparation. Il n'a pas été sous-estimé en terme de planning. Il a été validé au préalable par un ensemble de personnes. Cependant, les délais proposés pour chacune des tâches n'envisageaient pas de gros écart en cas d'imprévus.

5.2 Les perspectives d’évolutions

Il reste plusieurs étapes à réaliser pour finaliser ce projet et l’installer en production définitivement. J’ai réparties ces étapes dans le temps pour les réaliser dans les meilleures conditions.

La première étape consiste à l’installation du système en parallèle du système existant. Cette installation se faisant sur des nouveaux serveurs, je pourrais réaliser et finaliser l’ensemble des tests non effectués jusqu'à présent tels que les tests de performances. Ces tests me permettront de dimensionner le type et le nombre de serveur à mettre en place pour un fonctionnement optimal.

La seconde étape correspond aux portages de l’ensemble des applications non traitées en relation avec la base actuelle pour quelles utilisent la nouvelle structure (services web).

La troisième étape permet, après validation de la maîtrise d’ouvrage, l’arrêt du système existant pour l’utilisation exclusive du nouveau système.

Le projet étant orienté avec les données de la ligne A, la dernière étape consiste à réaliser les mêmes opérations pour la ligne B. Il faut intégrer l’ensemble des données de cette ligne dans la base de données puis effectuer les mêmes opérations de déploiement sur le système existant de la ligne B qui est composé pour rappel des mêmes éléments que la ligne A.

De plus, une application est en cours de portage pour une gestion dynamique des données en utilisant les services web. Elle remplacer l’application GENDON qui génère aujourd'hui les données sous forme de fichiers. Par la même occasion, on supprimera les scripts et les fichiers SQL de transfert qui permettait de faire temporairement le transit entre les données fichiers et la nouvelle structure.

Pour finir, la base va s’étoffer de nouvelles données. Celles des lignes de Métro et du Tramway vont être insérées. Elles permettront à l'avenir de fournir les mêmes services que le RER. Les clients lourds sont beaucoup plus nombreux sur le Métro que sur le RER. L'un de leurs inconvénients actuels vient du fait qu'ils sont développés sous Windows qui nous pose des problèmes de compatibilité.

6 Synthèse de la formation

Suite à la fin de mon DUT et de ma licence informatique en alternance, je souhaitais à tout prix quitter l'école pour le monde de l'entreprise. L'alternance favorise les étudiants à appréhender le monde de l'entreprise plus rapidement. J'ai été embauché en 2007 à INEO puis à la RATP en 2008 en tant qu'analyste programmeur.

Après un an et quelques regrets, je souhaitais reprendre mes études pour faire évoluer mon poste. J'ai recherché les différentes possibilités de formation. La RATP ne les faisait plus en interne. Les cours du soir restaient la seule solution étant dans l'impossibilité d'arrêter mon travail.

Le CNAM est spécialisé dans la formation en cours du soir et propose un panel très varié de cours dans beaucoup de disciplines.

Mon principal objectif était de rester dans le domaine de la gestion de projet informatique. Le CNAM propose le diplôme d'ingénieur avec l'option architecture et ingénierie des systèmes et des logiciels (AISL). Il correspond à ce que je souhaite par la suite. Je m'inscris en 2009 sur ce diplôme.
Après six années, la formation m'a permis de suivre un ensemble de cours basés sur de la technique, de la comptabilité et de la gestion de projet. Ces cours présentent des méthodes et des règles à suivre pour nous améliorer et nous orienter dans les bonnes directions. En finalité, la formation nous rend compétent dans plusieurs domaines.

Cette formation m'a permis de rencontrer des enseignants chercheurs du CNAM et d'entreprises extérieures qui m'ont apporté beaucoup en termes de connaissances et de valeurs.

Je suis très content d'avoir suivi les cours du soir. Cette formule m'a permis de progresser professionnellement mais aussi j'ai pu me former en même temps ce qui a été très enrichissant.

Conclusion

Ce projet permet d'ouvrir la porte à de nouvelles perspectives dans le partage d'information. Cependant, il est loin d'être finalisé dans son ensemble, il reste encore beaucoup de travail.

Une répartition de la charge restante est en cours de délibération avec ma hiérarchie. Elle consiste à répartir les tâches des modules restants mais aussi d'effectuer un travail en binôme. Cette méthode de travail fait partie de notre quotidien. Il est indispensable pour notre service de maintenance de partager la connaissance sur les projets. En cas d'absence du principal intervenant, le second est opérationnel lors de dysfonctionnements.

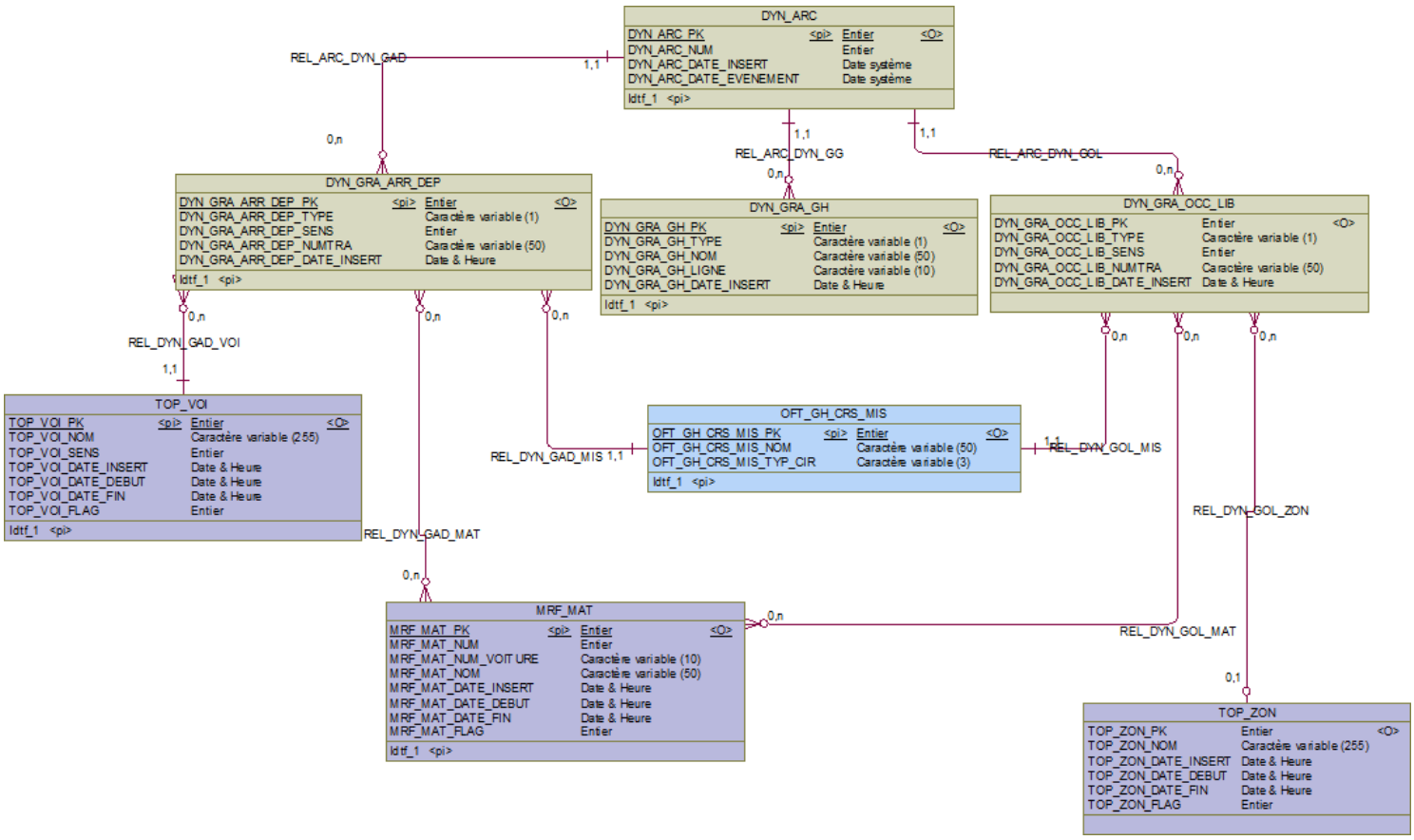
Pour le projet, bien qu'il y ait des préconisations, j'ai eu la chance d'avoir une entière liberté sur les méthodes et les outils employés. Les idées et leurs mises en œuvre se sont basées sur l'expérience professionnelle acquise sur mes projets mais aussi le retour d'expérience (REX) sur les projets déjà réalisés dans notre entité. De nombreuses recherches ont permis aussi d'approfondir et de fonder ce qui a été appliqué.

L'ensemble des actions effectuées sur ce projet, m'a fait acquérir de nouvelles compétences dans le domaine de la base de données et les méthodes de gestion pour réussir un projet dans les meilleures conditions.

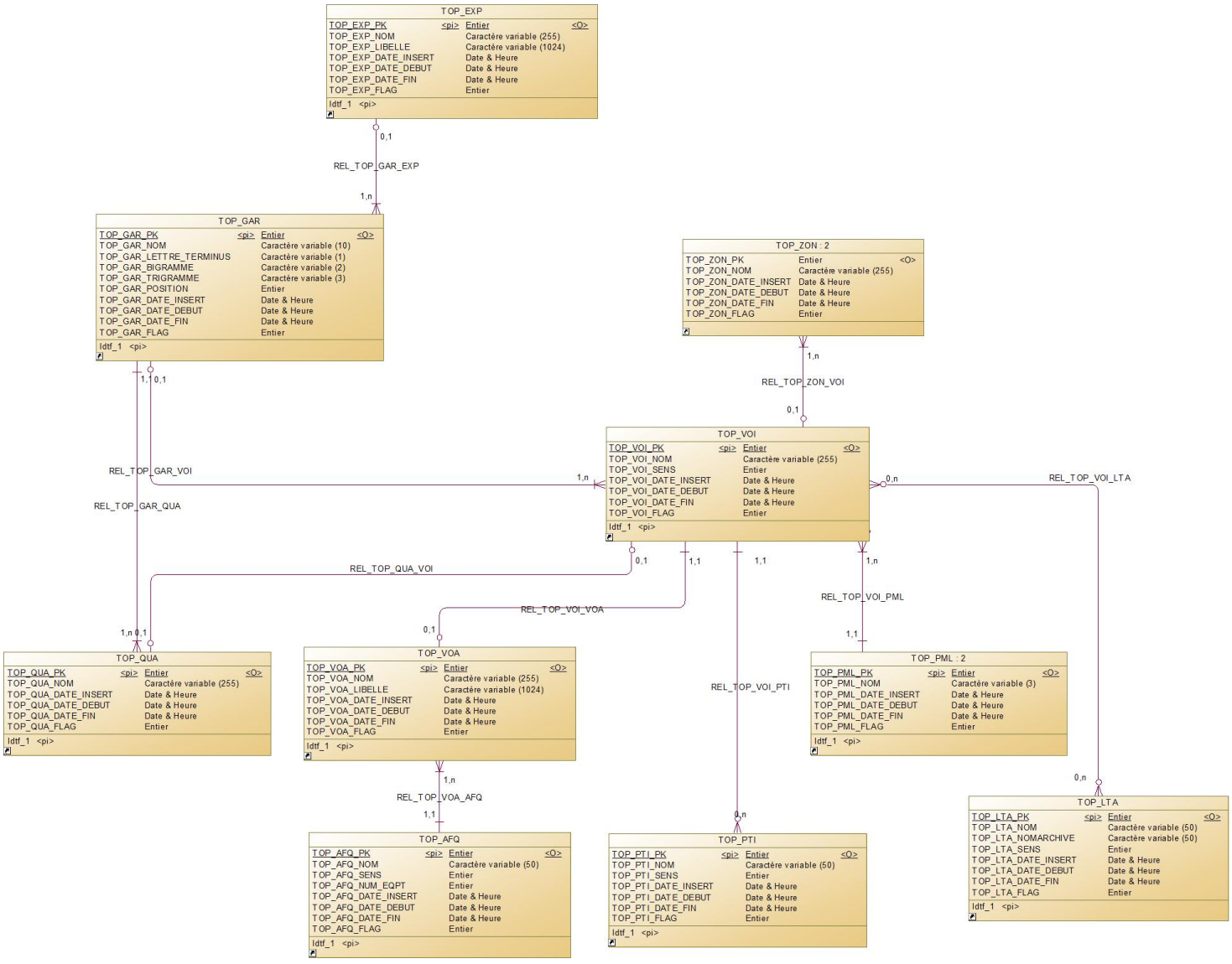
Le projet vient clôturer six années de cours du soir dans l'optique d'obtenir le diplôme d'ingénieur. Ce diplôme me permettra d'évoluer à la RATP dans des postes d'encadrement dans la gestion de projet informatique.

Annexes

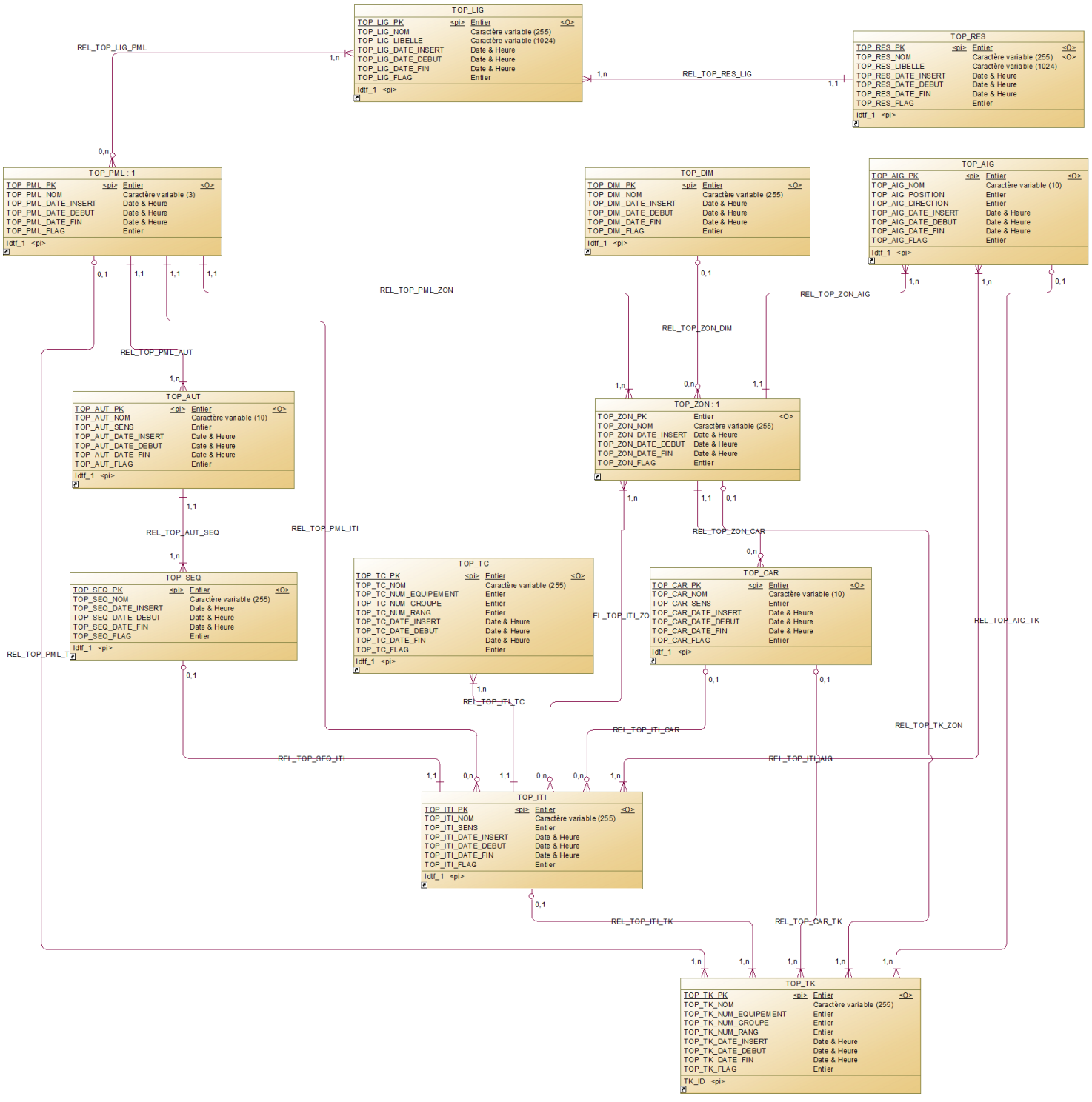
Annexe 1 : Modélisation conceptuelle des tables de l'archivage



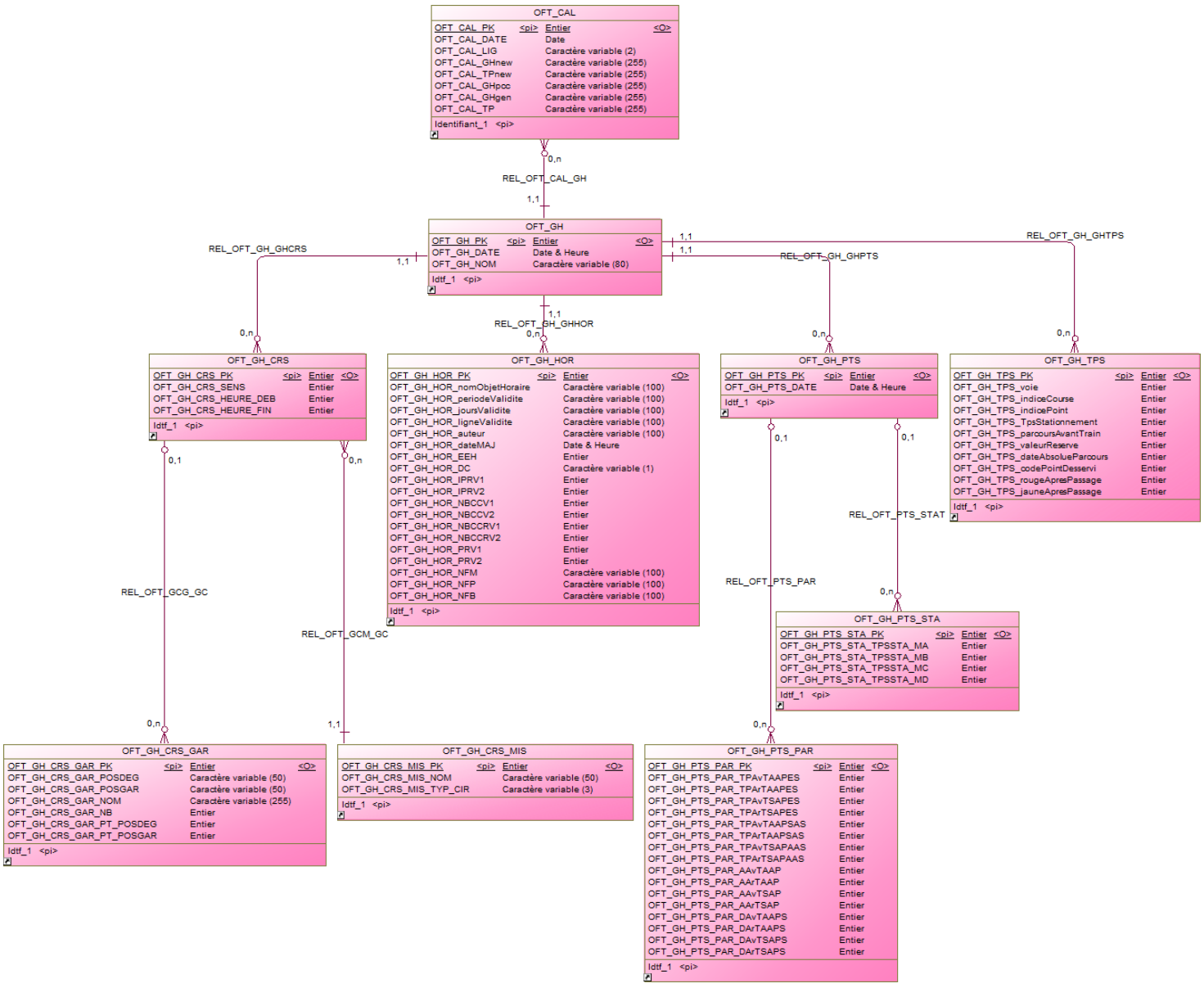
Annexe 2 : Modélisation conceptuelle de la topologie (1/2)



Annexe 3 : Modélisation conceptuelle de la topologie (2/2)



Annexe 4 : Modélisation conceptuelle de l'offre de transport



Bibliographie

[1] LONDON UNITED, 2009. [En ligne]. Url : https://www.ratpdev.com/fr/london-united, [Consulté en 04/2015].

[2] RATP DEV BRASIL SÃO PAULO, 2006. [En ligne]. Url : https://www.ratpdev.com/fr/ratp-dev-brasil-sao-paulo, [Consulté en 04/2015].

[3] Préconisations RATP, 03/2015 « CCTG-FRI (Cahier des Clauses Techniques Générales pour les marchés de Fourniture et de Réalisation Informatique) » [En ligne] Url : http://urbanweb.info.ratp/urbanweb/upload/docs/application/pdf/2013-02/mig080627_cctg-fri.pdf [Consulté en 04/2015]

[4] Type de réplication, 2015. « Haute disponibilité, répartition de charge et réplication». [En ligne]. Url : http://docs.postgresqlfr.org/9.2/high-availability.html, [Consulté en 01/2015].

[5] Réplication, 02/2006, « Introduction à la réplication de bases de données » [En ligne] Url : http://greg.rubyfr.net/pub/?page_id=20 [Consulté en 04/2015]

[6] Services web, 01/2015, [En ligne] Url : https://fr.wikipedia.org/wiki/Service_web [Consulté en 02/2015]

[7] Web services, 2003, [En ligne] Url : http://deptinfo.cnam.fr/Enseignement/CycleSpecialisation/ISA/ISCS/Laloum/WEB_Services.pdf [Consulté en 2015]

[8] Architecture REST, 2006, [En ligne] Url : http://www.figer.com/publications/REST.htm#.VfK4hdBOJDA [Consulté en 2015]

[9] TCP/IP, Sami Taktak, «NFA083 – Réseau et Administration Web TCP/IP » [En ligne] Url : http://cedric.cnam.fr/~taktaks/NFA083/TCPIP.pdf [Consulté en 2015]

[10] Informix, 27/09/2014. [En ligne]. Url : http://fr.wikipedia.org/wiki/Informix, [Consulté en 04/2015].

[11] Poussineau J., 24/06/2011. « La performance des applications ». [En ligne]. Url : https://blog.axopen.com/2011/06/la-performance-des-applications, [Consulté en 04/2015].

[12] Laloum Y., 2013. « Contrat de service et gestion de l'exploitation ». Paris, 75. [Cours suivi en 2013 – GLG102].

[13] Brouard F., 27/04/2012. « Base de données et performances… petites tables et tables obèses ! ». [En ligne]. Url : http://blog.developpez.com/sqlpro/p10070/langage-sql-norme/base_de_donnees_et_performances_petites, [Consulté en 02/2015].

[14] Yann Lemaulf, 2011, [En ligne] Url : http://igm.univ-mlv.fr/~dr/XPOSE2011/choisirsabasededonnee/disponibilite.html [Consulté en 2015]

[15] turblog, 24/11/2010. « La redondance». [En ligne]. Url : https://blog.spyou.org/wordpress-mu/2010/11/24/la-redondance/, [Consulté en 04/2015].

Tables des illustrations

Figure

Tableau

RESUME

La RATP souhaite fournir à ses usagers un service de qualité. Pour le réalisé, les exploitants s'appuient sur des outils d'aide à l'exploitation. Ils peuvent être représentés sous la forme d'applications graphiques telles que des IHM. Elles simplifient et automatisent de nombreuses tâches qui peuvent être complexe.

La conception de ces applications a été réalisée indépendamment les unes des autres ce qui posent plusieurs problèmes. On se retrouve avec des données dédoublées avec un risque d'oubli lors de modification ou d'évolution de ces données. De plus, nous possédons une base de données actuelle qui a été mal conçue et sous-estimé, ne pouvant plus fournir les besoins d'aujourd'hui.

Pour nous assurer d'une meilleure gestion des données, ce mémoire consiste à mettre en place les bases d'une nouvelle architecture matérielle et logicielle. La conception de la base est revue de zéro. On en profite pour modifier la méthode d'accès aux données (service web) et porter une application de type client lourd en client léger.

Mots clés

Base de données – Services web – Client léger – Maintenance

SUMMARY

RATP wants to provide its customers with quality service. For achieved, operators rely on the operating aids. They can be represented as graphics applications such as HMI. They simplify and automate many tasks that can be complex.

The design of these applications was carried out independently of each other which are several problems. We end up with duplicated data with a risk of forgetting when modification or evolution of such data. In addition, we have a current database that was poorly designed and underestimated, unable to provide the needs of today.

For us assured of better data management, this thesis is to establish the foundations of a new hardware and software architecture. The database design is reviewed from scratch. We took the opportunity to change the data access method (web service) and wear a thick-client application in thin client.

Key words

Database - Web Services(Departments) - light Customer - Maintenance

FIN DU DOCUMENT