

http://www.google.fr/url?source=imglanding&ct=img&q=http://www.smart-trade.net/wp-content/uploads/2012/10/logo2.png&sa=X&ei=A-FdVaqJD8z6UIjLgcAB&ved=0CAkQ8wc&usg=AFQjCNHYCIWvqf67IeNkykCIdYErgRsn2g



Seddik Ouiss

Smart trade technology

Rapport de stage

Master Intégration de systèmes logiciels.

**CLAUSE DE CONFIDENTIALITÉ**

Les informations contenues dans ce document sont confidentielles et ne doivent en aucun cas être copiées ou diffusées sans l’autorisation de Smart Trade Technologies.

# Remercîment

Tout d'abord, je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce rapport.

Je tiens à remercier vivement mon maitre de stage, Mr Julien Aymard, responsable au sein de Smart-Trade Technologies du projet R5, pour son accueil, le temps passé ensemble et le partage de son expertise au quotidien. Grâce aussi à sa confiance, j'ai pu m'accomplir totalement dans mes missions avec son aide précieuse dans les moments les plus délicats.

Je remercie également à toute l’équipe R5, pour le temps passé à m’expliquer le travail déjà réalisé dans le projet R5, me facilitant ainsi l’intégration dans leur équipe.

Je remercie également Mr Eric Deshayes, responsable à de Smart-Trade Technologies, pour ces directives techniques ainsi que ses conseils sur l’intégration de mon sujet dans la plateforme technique de Smart Trade.

J’adresse mes remerciements à mon professeur, Mr Jean-Luc Massat, mon encadrant, ainsi que tous les professeurs de l'Aix-Marseille Université qui nous ont enseignés et partagés leurs connaissances durant mon cursus.

Je remercie bien-sûr M. David VINCENT pour m’avoir accepté dans son entreprise, ainsi que l’ensemble des employés de Smart Trade Technologies pour l’aide, le soutient et l’ambiance de travail chaleureuse qu’ils m’ont apportés.

Enfin, je tiens à remercier toutes les autres personnes qui m'ont conseillées et aidées et soutenues durant toute la durée de ce stage : ma famille, mes amis, mes camarades proches.

|  |  |
| --- | --- |
| Etudiant : | Seddik Ouiss |
| Année : | 2015 |
| Raison social de l’entreprise : | Smart Trade Technologies |
| Tuteur de stage : | Julien Aymard |
| Enseignant responsable : | Jean-Luc Massat |
| Sujet du stage : | L’objectif du stage est de mettre en place une infrastructure facilitant le développement d’une application de *reporting*.  Cette dernière sera utilisée par les décideurs chez les clients. |
| Plate-forme-informatique : | Environnement Smart Trade (conteneur léger basé sur Spring) |
| Outils & logiciels : | Eclipse, Java, Spring, PonySDK(Basé sur GWT), Pivot4J, Olap4J, Mondrian, Hibernate, Pentaho Kettle, Birt, JasperSoft, Pentaho Report Designer |

# Sommaire :

[Remercîment 2](#_Toc422236888)

[Sommaire : 4](#_Toc422236889)

[Introduction : 5](#_Toc422236890)

[Présentation de l’entreprise 6](#_Toc422236891)

[Travail réalisé : 7](#_Toc422236892)

[Introduction : 7](#_Toc422236893)

[Objectif et travail demandé : 7](#_Toc422236894)

[Logiciels, outils informatiques et protocoles utilisés : 7](#_Toc422236895)

[Etude préalable : 7](#_Toc422236896)

[Bilan du travail : 8](#_Toc422236897)

[Bilan du travail en entreprise : 8](#_Toc422236898)

[Bilan du travail personnel : 8](#_Toc422236899)

[Bilan de la formation : 8](#_Toc422236900)

[Conclusion 9](#_Toc422236901)

[Bibliographie 10](#_Toc422236902)

[Site web : 10](#_Toc422236903)

[Journal de stage : 10](#_Toc422236904)

# Introduction

Dans le cadre de l’obtention du Master 2 professionnel en Intégration de systèmes logiciels (ISL), la formation exige de chaque étudiant d’effectuer un stage de fin d’études idéalement dans le domaine qui l’intéresse pour une durée de 5 à 6 mois.

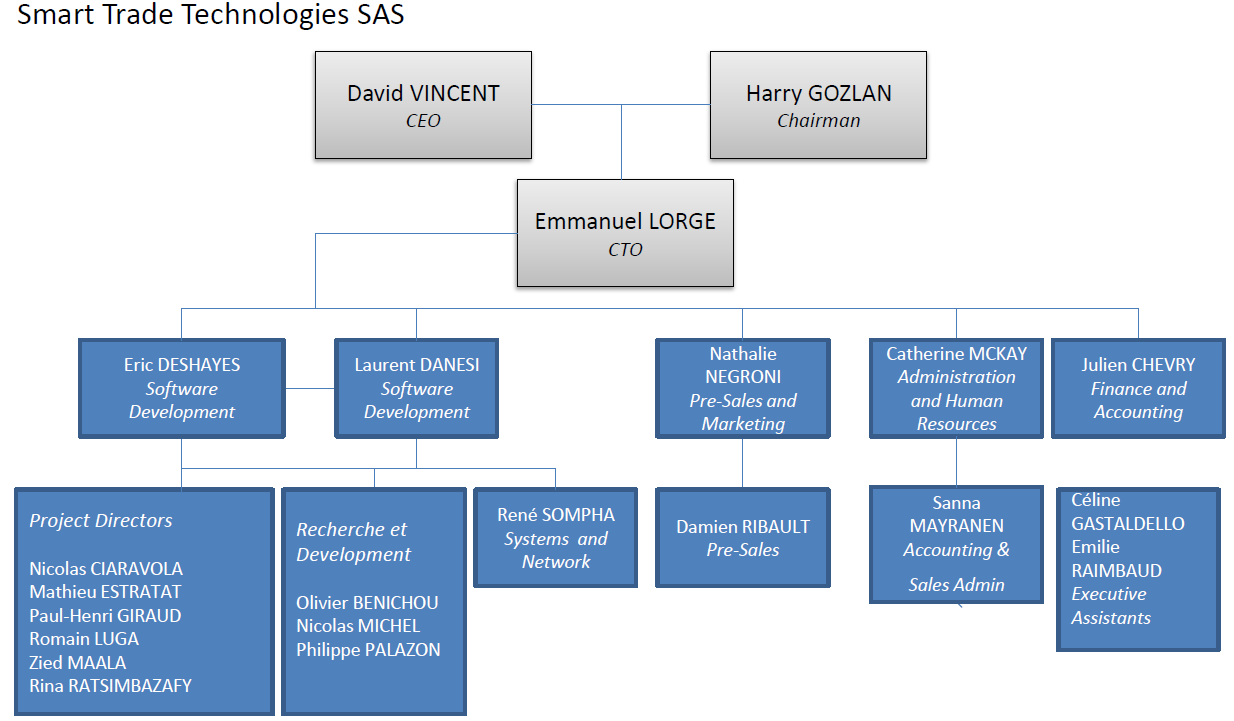
Désirant mettre en pratique les connaissances acquises durant ma formation, j’ai ressenti le besoin de m’orienter vers une entreprise dont la rigueur et la qualité sont les principales priorités. J’ai débuté mes recherches auprès des entreprises du secteur dès octobre 2014. Elles ont abouti au mois de décembre lorsque l’entreprise Smart Trade Technologies a accepté de m’accueillir en tant que stagiaire d’Avril à Septembre 2015.

L’objectif de mon stage était d’ajouter à Smart Trade la possibilité de faire du « Business Intelligence » ou plus précisément du « reporting », c’est-à-dire permettre aux clients Smart Trade d’exploiter l’énorme masse de donné stocker via les outils Smart Trade et de pouvoir l’utiliser à des fins commerciales ou stratégiques.

Pour ce faire, j’ai dû étudier ce qui se faisait déjà chez Smart Trade et ensuite ce qui se faisait de mieux dans ce domaine. Ceci m’a amené à une phase d’étude préalable dans laquelle j’ai réalisé plusieurs prototypes techniques qui ont permis de délimiter le sujet de stage.

Dans ce rapport, après la présentation de l’entreprise et des objectifs principaux de mon stage, j’aborderai l’essentiel du travail réalisé et quelques cas d’utilisation.

# Présentation de l’entreprise

SmartTrade Technologies fut créée en 1999, par Harry GOZLAN qui occupe aujourd’hui le poste de président exécutif au sein de l’entreprise, et David VINCENT au poste de CEO 2. Smart Trade fournit des systèmes de gestion de la liquidité sophistiqués, permettant aux banques de développer des plateformes de transaction multi-actifs. Ces systèmes regroupent des liquidités de dizaines de sources pour créer un carnet d’ordres uniques, de distribuer des prix personnalisés aux clients. De recevoir et de gérer les flux d’ordres de clients, internaliser des liquidités aux sites externes. Ils gèrent l’état du cycle de vie de l’ordre, en mettant en œuvre le routage intelligent des ordres et un des algorithmes propriétaires, en reliant électroniquement à un certain nombre de lieux, indépendamment des protocoles de messagerie. SmartTrade est implantée dans plusieurs villes à travers le monde afin d’être plus près de ses clients, à savoir Londres, New York, Tokyo, et Aix en Provence où se déroulé mon stage comportant plus d’une quarantaine de personnes travaillant sur différents projets. 

Smart Trade propose aussi un LMS (Liquidity Management System) sophistiqué, permettant aux banques :

* La gestion de leurs liquidités.
* L’optimisation des processus de trading.
* La gestion des différentes règles de management de prix entre les ventes.
* Faire gagner du temps aux traders.
* Faire des trades sur tout type d’instruments (matières première, monnaie, actions, etc.).

Organisation des équipes à Aix en Provence

– CORE : fournit tous les éléments nécessaires pour le côté backend, c’est-à-dire gère l’ensemble des bibliothèques de tous les composants métiers de smartTrade.

– CORE UI : fournit tous les éléments nécessaires pour le côté frontend, c’est-à-dire gère l’ensemble des bibliothèques des tous les composants graphiques de smartTrade.

– LFX : LFX 3 étant un produit smartTrade, cette équipe s’occupe essentiellement de l’amélioration du produit.

– QA : contrôle la qualité des logiciels de smartTrade du point de vue fonctionnel. – Bench : contrôle la capacité et les ressources utilisées par les logiciels afin de les optimiser

## Organisation des équipes à Smart Trade

A Smart Trade il y a plusieurs équipes comme le montre le schéma ci-dessous ou on voit l’équipe où je suis affecté c’est CORE UI elle fournit tous les éléments nécessaires pour le côté frontend, c’est-à-dire gère l’ensemble des bibliothèques des tous les composants graphiques de smartTrade.

Figure 2 : la liste des projets smartTrade

## Outils et services

Pour le bon déroulement du travail et pour produire un résultat de qualité l’entreprise nous mis à disposition plusieurs outils permettant de simplifier le travail qui sont :

* TimeSheet : comme outil de gestion des ressources, développé en interne.
* Jenkins : pour les tests intégration continues.
* CVS et GitLab : pour la gestion de versions des projets Smart Trade.
* Confluence : la base de connaissances de l’entreprise.
* Artifactory : la base de données des librairies internes et externes utilisées dans l’entreprise.

Et pour faciliter la communication au sein de l’entreprise, et avec les clients, les outils ci-dessous sont aussi utilisés :

* JIRA : système de suivi de bug, de gestion des incidents et de gestion de projets.
* Confluence : la base de connaissances de l’entreprise.
* Skype et Hangout : messagerie instantanée.
* mais aussi les mails, fax, et téléphone.

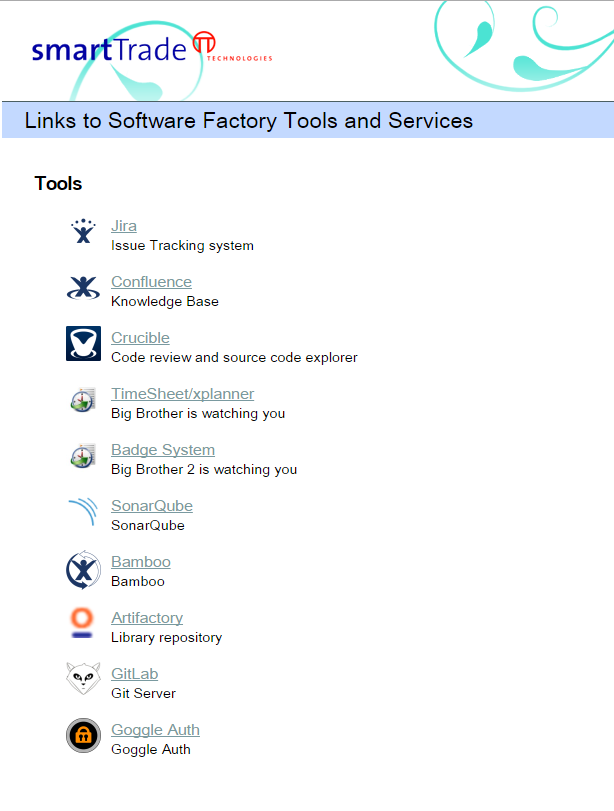


Figure 3: Outils et services smartTrades

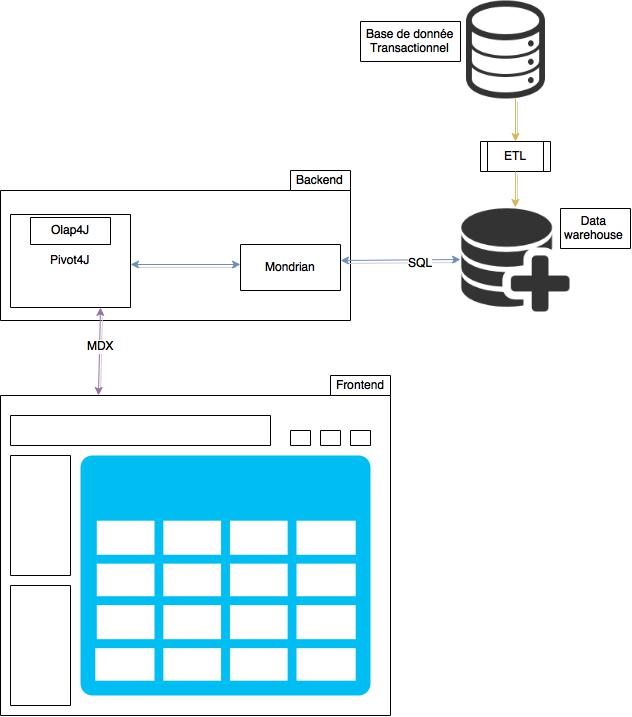
# Travail réalisé

## Introduction :

L’objectif est de mettre en place une infrastructure facilitant le développement d’une application de *reporting*.

Cette dernière sera utilisée par les décideurs chez les clients.

## Infrastructure :



* Pour faire cette application, on peut mettre en place une ETL qui va agréger les différentes sources de données (Big Data, Base de données SQL, Fichier transactionnels, etc...) dans un entrepôt de données. Le fait de mettre en place un entrepôt de données permettra :
* D’avoir une source de données assez grande pour des statistiques plus fiables.
* De décharger les différentes sources de données liées aux transactions fonctionnelles des requêtes de reporting.
* De faire des prédictions.
* Ensuite on pourra mettre en place un serveur OLAP qui permettra de faire des requêtes sur l’entrepôt de données.

Une API java du nom d’Olap4j permet d’interagir avec un serveur OLAP.

A partir de l’application Back-end, il sera possible de faire des requêtes statiques (ex : le trader qui a vendu le plus ce dernier mois), des requêtes paramétrées (ex : les ventes d’un trader entre 2014 et 2015) ainsi que des requêtes dynamiques.

Les requêtes dynamiques seront déterminées par les décideurs chez le client.

L’application Front-End, se présentera en quatre onglets :

* le premier présentera le résultat des requêtes statiques.
* le second affichera le résultat de requêtes paramétrées (avec la possibilité de les paramétrer).
* le troisième montrera une liste de valeur (correspondante à des colonnes des différentes sources de données ainsi que les fonctions : AVG, COUNT, SUM, etc…). Ces différentes valeurs pourront être déplacées (drag & drop) pour mettre en place des tableaux croisés. On pourra utiliser Pivot4J qui permet de faire des tableaux croisé.

## Objectif et travail demandé :

## Logiciels, outils informatiques et protocoles utilisés :

* Java :

Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.

La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java.

La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d’exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. Pour cela, divers plateformes et frameworks associés visent à guider, sinon garantir, cette portabilité des applications développées en Java.

* Highcharts :

Permet de créer des diagrammes interactifs facilement pour vos projets web.

Utilisé par des dizaines de milliers de développeurs et 61 des 100 plus grandes entreprises du monde, Highcharts est le plus simple API de diagrammes encore plus flexible sur le marché.

* JavaScript :

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs. Il a été créé en 1995. Il peut être directement intégrer au sein des pages web, pour y être exécuté sur le poste client. C'est alors le navigateur Web qui prend en charge l'exécution de ces programmes appelés scripts.

* PonySDK :

Est un Framework open source cote front end qui englobe GWT qui permet de créer des applications web rapidement en utilisant le langage java, développer par : Nicolas Ciaravola, Mathieu Barbier, Luciano Broussal.

* Spring

Spring est un socle pour le développement d'applications, principalement d'entreprises mais pas obligatoirement. Il fournit de nombreuses fonctionnalités parfois redondantes ou qui peuvent être configurées ou utilisées de plusieurs manières : ceci laisse le choix au développeur d'utiliser la solution qui lui convient le mieux et/ou qui répond aux besoins.

Spring est ainsi un des frameworks les plus répandus dans le monde Java : sa popularité a grandie au profit de la complexité de Java EE notamment pour ses versions antérieures à la version 5 mais aussi grâce à la qualité et la richesse des fonctionnalités qu'il propose :

* son cœur reposant sur un conteneur de type IoC (Inversion of Control) assure la gestion du cycle de vies des beans et l'injection des dépendances
* l'utilisation de l'AOP (programmation orienté aspect)
* des projets pour faciliter l'intégration avec de nombreux projets open source ou API de Java EE

Spring était un framework applicatif à ses débuts mais maintenant c'est une véritable plate-forme composée du framework Spring, de projets qui couvrent de nombreux besoins et de middlewares.

Spring permet une grande flexibilité dans les fonctionnalités et les projets utilisés dans une application. Il est par exemple possible d'utiliser le conteneur Spring pour gérer de façon basique les beans sans utiliser l'AOP. Par contre, certains projets et certaines fonctionnalités ont des dépendances avec d'autres projets.

Spring est associé à la notion de conteneur léger (lightweight container) par opposition aux conteneurs lourds que sont les serveurs d'applications Java EE.

* Git :

Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2.

* Eclipse

Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la Fondation Eclipse visant à développer un environnement de production de logiciels libre qui soit extensible, universel et polyvalent, en s'appuyant principalement sur Java.

Son objectif est de produire et fournir des outils pour la réalisation de logiciels, englobant les activités de programmation (notamment environnement de développement intégré et frameworks) mais aussi d'AGL recouvrant modélisation, conception, testing, gestion de configuration, reporting… Son EDI, partie intégrante du projet, vise notamment à supporter tout langage de programmation à l'instar de Microsoft Visual Studio.

Bien qu'Eclipse ait d'abord été conçu uniquement pour produire des environnements de développement, les utilisateurs et contributeurs se sont rapidement mis à réutiliser ses briques logicielles pour des applications clientes classiques. Cela a conduit à une extension du périmètre initial d'Eclipse à toute production de logiciel : c'est l'apparition du framework Eclipse RCP en 2004.

Figurant parmi les grandes réussites de l'Open source, Eclipse est devenu un standard du marché des logiciels de développement, intégré par de grands éditeurs logiciels et sociétés de services. Les logiciels commerciaux Lotus Notes 8, IBM Lotus Symphony ou WebSphere Studio Application Developer sont notamment basés sur Eclipse.

* CVS

CVS (sigle de Concurrent Versions System) est un système de gestion de versions créé en 1990, qui a été largement utilisé par les projets de logiciels libres.

Puisqu'il aide les sources à converger vers la même destination, on dira que CVS fait la gestion concurrente de versions ou de la gestion de versions concurrentes. Il peut aussi bien fonctionner en mode ligne de commande qu'à travers une interface graphique. Il se compose de modules clients et d'un ou plusieurs modules serveur pour les zones d'échanges.

Le premier logiciel qui a offert le service CVS s'appelle CVS, parmi les autres logiciels qui offrent un tel service il y a Cervisia, linCVS, TortoiseCVS, WinCVS, CVSNT.

* Gradle

Gradle est un moteur de production fonctionnant sur la plateforme Java. Il permet de construire des projets en Java, Scala, Groovy voire C++.

Gradle allie les atouts d’Apache Maven et d’Apache Ant : il allie l'utilisation de conventions à la manière de Maven (convention plutôt que configuration) avec la flexibilité d’Ant pour décrire les tâches de constructions, avec une cohérence forte dans l'interface de programmation des tâches.

* Mondrian

**Mondrian** est un moteur **OLAP** (Online Analytical Processing) écrit en Java par [Julian Hyde](http://julianhyde.blogspot.com/) qui permet la conception, la publication et le requêtage de cubes multidimensionnels.

Mondrian permet l’exécution de requêtes en langage MDX sur des entrepôts de données s’appuyant sur des SGBDR, d’où sa caractérisation de « ROLAP » (Relational OLAP). En matière de **ROLAP**, **Mondrian est la référence open source.**

* Mondrian permet d’accèder aux résultats dans un format multidimensionnel compréhensible par une API de présentation côté client, le plus souvent en mode Web, avec par exemple **JPivot**, **Pentaho Analyzer**, **Pentaho Analysis Tool**, **Geo Analysis Tool** **(G.A.T.)**
* Mondrian s’appuie sur une modélisation OLAP standard et peut donc se connecter à n’importe quel entrepôt de données conçu dans les règles de l’art de la Business Intelligence.
* Il est intéressant de noter que Mondrian est le composant OLAP utilisé par la plupart des suites de BI Open Source notamment **Pentaho**, **JasperServer** et **SpagoBI**.
* **Jedox PALO** fait d’ailleurs figure d’exception avec son moteur spécifique MOLAP (dans ce cas les données du cubes sont toutes précalculées, ce qui possède l’avantage d’être très rapide côté temps de réponse, mais un peu plus contraignant lorsqu’on souhaite modifier la vue OLAP ou accéder aux données de détail de la table de faits)
* **Pentaho** étant l’éditeur qui héberge et soutient la communauté Mondrian ([mondrian.pentaho.org](http://mondrian.pentaho.org)), je présente ci-dessous un schéma d’architecture de Mondrian et son utilisation au sein de la plate-forme Pentaho.

-Olap4J

Olap4j est une API ouverte Java pour accéder aux données OLAP.

Il est une extension de JDBC. Par exemple, sa classe de OlapConnection étend java.sql.Connection, à partir de laquelle vous pouvez créer une OlapStatement, et exécuter pour créer un ensemble de cellules (analogue à un java.sql.ResultSet). Il ya également des mécanismes similaires pour parcourant les métadonnées. En conséquence, olap4j est facile à apprendre si vous avez de l'expérience JDBC et savez un peu sur OLAP.

-Pivot4J

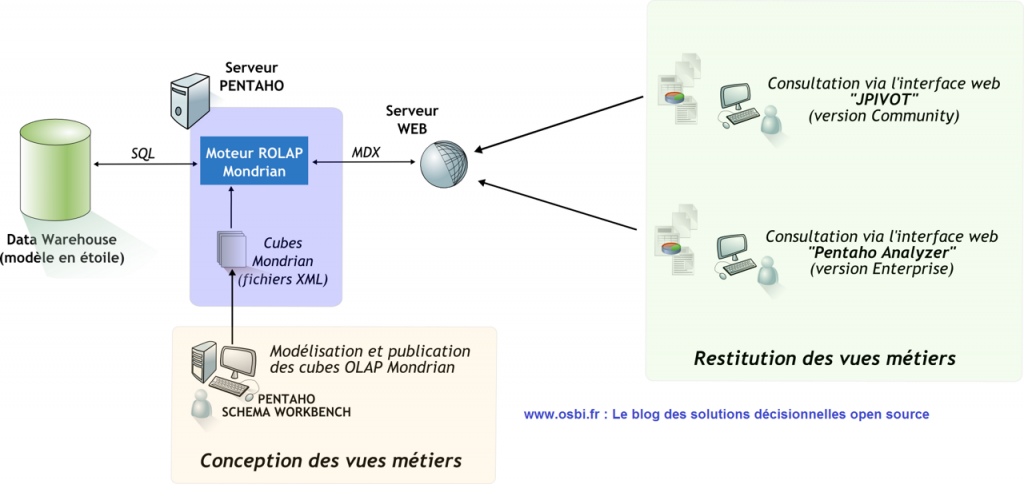
Pivot4J est une API Java pour les serveurs OLAP qui peuvent être utilisés pour construire un frontend service d'analyse avec une interface graphique de style de pivot . Il vise à tirer parti mature, mais maintenant discontinué la base de code de projet JPivot pour en faire un OLAP bibliothèque d'API d'usage général qui est indépendant de toute implémentation graphique particulier.

Pivot4J fournit également une application entièrement fonctionnelle d'analyse OLAP construit au-dessus de sa bibliothèque centrale , avec un plugin pour la plateforme Pentaho BI qui peut être installé directement sur ​​le marché.

-Jira

**JIRA** est un [système de suivi de bugs](https://fr.wikipedia.org/wiki/Logiciel_de_suivi_de_probl%C3%A8mes), un système de [gestion des incidents](https://fr.wikipedia.org/wiki/Gestion_des_incidents), et un système de [gestion de projets](https://fr.wikipedia.org/wiki/Gestion_de_projets) développé par [Atlassian Software Systems](https://fr.wikipedia.org/wiki/Atlassian_Software_Systems).

## Etude préalable :



<!-- Trades Cube -->

<Cube name=*"Trades"*>

<Table name=*"trade\_facts"* />

<DimensionUsage name=*"Product"* source=*"Product"*

foreignKey=*"id\_product"* />

<DimensionUsage name=*"Time"* source=*"Time"*

foreignKey=*"id\_time"* />

<Dimension name=*"Buyer"* foreignKey=*"id\_buyer"*>

<Hierarchy hasAll=*"true"* allMemberName=*"All Buyers"*

primaryKey=*"id\_buyer"*>

<Table name=*"dim\_buyer"* />

<Level name=*"Counter"* column=*"counter"* uniqueMembers=*"true"* />

<Level name=*"Name"* column=*"name"* uniqueMembers=*"true"* />

</Hierarchy>

</Dimension>

<Dimension name=*"Seller"* foreignKey=*"id\_seller"*>

<Hierarchy hasAll=*"true"* allMemberName=*"All Sellers"*

primaryKey=*"id\_seller"*>

<Table name=*"dim\_seller"* />

<Level name=*"Counter"* column=*"counter"* uniqueMembers=*"true"* />

<Level name=*"Name"* column=*"name"* uniqueMembers=*"true"* />

</Hierarchy>

</Dimension>

<Dimension name=*"Order Buy"* foreignKey=*"id\_order\_buy"*>

<Hierarchy hasAll=*"true"* allMemberName=*"All Order Buy"*

primaryKey=*"id\_order\_buy"*>

<Table name=*"dim\_order\_buy"* />

<Level name=*"Order"* column=*"order"* uniqueMembers=*"true"* />

</Hierarchy>

</Dimension>

<Dimension name=*"Order Sell"* foreignKey=*"id\_order\_sell"*>

<Hierarchy hasAll=*"true"* allMemberName=*"All Order Sell"*

primaryKey=*"id\_order\_sell"*>

<Table name=*"dim\_order\_sell"* />

<Level name=*"Order"* column=*"order"* uniqueMembers=*"true"* />

</Hierarchy>

</Dimension>

<Measure name=*"Trade Volume"* column=*"trade\_volume"* aggregator=*"sum"*

formatString=*"$ ###,###,###,##0.00"* />

<Measure name=*"Quantity"* column=*"quantity"* aggregator=*"sum"*

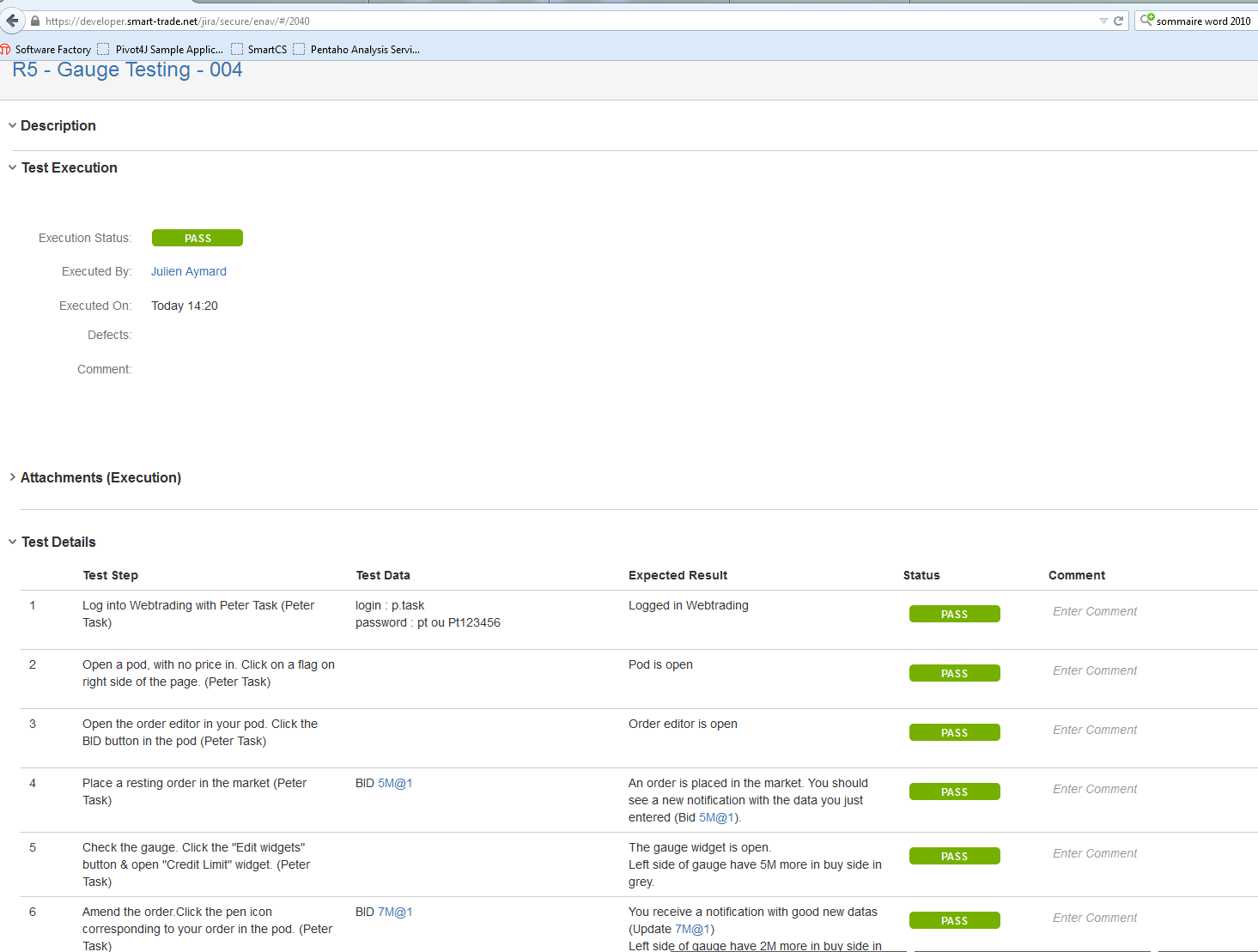
formatString=*"Standard"* />

<Measure name=*"Units Trade"* column=*"id"* aggregator=*"distinct-count"*

formatString=*"Standard"* />

</Cube>

# 



# Bilan du travail :

## Bilan du travail en entreprise :

## Bilan du travail personnel :

## Bilan de la formation :

# Conclusion

# 

# Glossaire

**Instrument :** C’est un produit sur lequel s’effectue une transaction. Exemple : actions, obligations, monnaie, etc.

**Trader :** C’est une personne, travaillant généralement pour une institution financière, spécialisé dans l’achat et la vente d’instrument.

**Ordre :** Un ordre est une requête envoyé par un trader sur le marché dans laquelle il spécifie le prix, la quantité, le coté (achat ou vente) et l’instrument entre autre.

**Trade :** C’est lorsque deux ordres opposés (achat et vente) se correspondent. A partir de cela, la transaction s’effectue entre les deux émetteurs de ces ordres.

# Bibliographie

# Journal de stage :