Haute Ecole LÉONARD DE VINCI
INSTITUT PAUL LAMBIN
Section Informatique

Université Catholique de Louvain

CONCEPTION D'OUTILS d'intégration automatisée pour ERP

MISE EN PLACE DE L'INFRASTRUCTURE d'une plate-forme d'e-Commerce

---

MÉMOIRE presenté par
FRYDMAN MAXIME
en vue de l'obtention du DIPLÔME de
BACHELIER en INFORMATIQUE de GESTION

Année académique 2011-2012

CONCEPTION D'OUTILS
d'intégration automatisée pour ERP

Mon stage a été accompli au sein de la compagnie:

BIRCHMAN CONSULTING
Avinguda Diagonal Mar 67
08019 BARCELONA.

MISE EN PLACE DE L'INFRASTRUCTURE
d'une plate-forme d' e-Commerce

Le client pour qui a été réalisé le projet:

Central TRUCCO: InSitu SA.
Calle Valportillo Primera 2
28108 ALCOBENDAS (Madrid)

Remerciements

Je tiens à remercier en particulier monsieur Joaquim Moya pour l'accueil chaleureux qu'il m'a réservé au sein de la compagnie BIRCHMAN.

Je suis très reconnaissant à mes professeurs de l'Institut Paul Lambin pour l'excellente formation qu'ils m'ont prodigué.

Un grand merci à madame Brigitte Lehmann pour ses conseils avisés pendant toute la rédaction de mon Travail Fin d'Études.

Merci de tout coeur à mes parents pour leur soutien et leur présence qui m'ont permis de faire mon stage dans d'excellentes conditions.

birchman

straightforward

clearthinking

tailoredoutcomes

Le *'Groupe'* BIRCHMAN est devenu un des principaux consultants au monde dans le domaine de la gestion des valeurs adaptées aux exigences particulières d'organisations individuelles. À l'aide de modèles opérationnels, les entreprises peuvent rationaliser leurs opérations et réduire leurs coûts. En cinq ans, *Birchman Group* est devenu une organisation mondiale avec des bureaux en Europe, en Amérique Latine, au Moyen-Orient, en Extrême-Orient et en Australie. La compagnie a récemment ouvert de nouvelles succursales en Afrique du Sud et en Écosse. [1]

Aujourd'hui, des entreprises à travers le monde, font face à des conditions économiques non rencontrées depuis des générations. Garantir l'optimalisation des frais est un des éléments les plus importants pour survivre à la crise.

Réduire les frais est relativement facile, mais les réduire efficacement pour garantir un avenir durable et optimal, ne l'est pas. L'histoire témoigne de nombreux exemples d'entreprises qui ont finalement échoué, pas à cause de la crise économique ou de la crise du marché, mais comme une conséquence directe de leur gestion. Le Groupe Birchman se concentre sur l'efficacité de la gestion des coûts avec une approche basée sur l'éthique et une méthodologie rigoureuse.

Dans le climat économique actuel, livrer des projets justes, meilleurs, plus rapides et moins chers que la concurrence créera, sans aucun doute, un avantage compétitif considérable. Dans ces circonstances, le Groupe Birchman est en train de devenir le **leader mondial** en ce qui concerne la consultance et le management. [2]

---

[1] La présentation de la société est basée sur le site Web du *Birchman Group*. Je me suis limité à la traduction des éléments qui sont explicitement liées à mon travail. Sur Birchman n'existe aucune littérature en langue française.

[2] STEPHENS K. Cost Effectiveness – Surviving and Thriving in the New Economic Order (2009) – http://www.the-financedirector.com/features/feature55231/

BIRCHMAN a été un partenaire **SAP** en Espagne depuis plus de 20 ans et a aidé des centaines d'entreprises à améliorer leurs capacités commerciales avec les solutions **SAP**. À l'aide d'un *service clients*, allant de conglomérats internationaux jusqu'à des petites entreprises, tous sont suivis par des équipes personnalisées qui connaissent leurs clients.

Les services informatiques ont la responsabilité de maintenir la fiable et robuste plate-forme **SAP** dont les entreprises d'aujourd'hui ont besoin. Dans le même temps, ils doivent étudier et mettre en œuvre des projets permettant de répondre aux nouvelles exigences et aux nouveaux besoins de l'entreprise.

Malheureusement, les activités quotidiennes de soutien **SAP** provoquent souvent des confusions parmi les équipes informatiques internes, les empêchant de se concentrer sur des questions plus importantes qui pourraient avoir une incidence négative sur le succès d'une entreprise.

En assumant des tâches de soutien systématique, Birchman réalise un gain de temps et fournit une totale tranquillité d'esprit pour que les systèmes **SAP** soient exécutés de manière optimale et que les utilisateurs travaillent efficacement.

Si un service **SAP** complet est indispensable, dans les périodes de pointe, Birchman peut garantir sa capacité à réagir rapidement et d'une manière avisée par son soutien à la clientèle.

Birchman fournit des services de SAP pragmatiques et flexibles qui génèrent de la valeur ajoutée y compris, de bout en bout, les Services SAP Dirigés, les Projets et les Licences. Birchman a un solide héritage de SAP par le biais de l'acquisition de PLAUT CONSULTING au Royaume-Uni et en Espagne.

Le Groupe a connu une croissance rapide depuis 2003, c'est aujourd'hui une équipe de 750 consultants dans 9 pays. [3]

Hillarys · Church & Dwight Co., Inc. · Halewood International Ltd · Burberry · Desigual · Pepe Jeans London · Torres "Un gran vino nace y se hace en casa" · Dynacast Solutions Made Solid · B&W Bowers & Wilkins

[3] PMTHINK! CIO Value Measurement: European Market Alliance... (2005) – http://www.pmthink.com/labels/business-strategy.htm

BIRCHMAN: VALUES - PRINCIPLES - CULTURE

'To be the world's leading outcomes based Consulting, Solutions and Managed Services Firm' Birchman sait se différencier grâce à sa focalisation sur une stratégie axée sur les objectifs particuliers de ses clients. Cette orientation permet au groupe de continuer à croître et à soutenir ses clients dans le monde entier.

BIRCHMAN: CORPORATE SOCIAL RESPONSIBILITY

Birchman tente de répondre de manière équilibrée aux besoins commerciaux, humains et de développement des clients et de l'environnement.

En travaillant en collaboration avec plusieurs universités réputées dans ce domaine, Birchman développe des méthodes et des outils de travail en matière de RSC. Ainsi a été mis au point un cadre à quatre dimensions qui s'applique chez les clients par des engagements consultatifs et par la pratique au sein de l'organisation Birchman.

TRUCCO **Fashion for Real Women**

TRUCCO appartient à la compagnie espagnole **InSitu** qui est établie à Madrid. Ils sont fabricant, distributeur, responsable de la conception, la fabrication et la commercialisation des collections pour TRUCCO. InSitu se spécialise dans la mode féminine depuis 20 ans. Hormis la création de collections pour sa propre marque, In Situ conçoit et fabrique des vêtements pour d'autres grands détaillants, produisant ainsi plus de 2.000.000 d'articles par an.

Avec plus de 25 ans d'expérience dans le secteur textile, TRUCCO offre du travail à plus de 450 personnes et à environ 200 points de vente dans le monde: L'Espagne, le Portugal, la République Tchèque, la Chine, l'Arabie Saoudite, le Mexique, le Qatar, le Singapour, le Panama, la Thaïlande, la République Dominicaine, le Guatemala, l'Équateur, la Costa Rica et l'Iran.

IP – e-COMMERCE

Depuis quelques années, les magasins TRUCCO ont été équipés avec des systèmes de surveillance vidéo et de comptage de personnes par la société MirameNET.

Cependant la recherche continue et de nouvelles solutions comme **l'intégration de données SAP dans les plates-formes de gestion** sont mis en place par BIRCHMAN.

C'est à ce développement que la seconde partie de mon projet est consacrée.

Le système de gestion est intégré verticalement, ce qui permet d'offrir au marché une grande souplesse et une excellente relation qualité-prix. TRUCCO a les moyens et les capacités de répondre de manière rapide et efficace, aux besoins de ses clientes étant capable de mettre un nouveau produit dans les magasins en moins de deux semaines.

Table des Matières

Introduction

Je suis arrivé dans l'entreprise peu après le début du projet. L'analyse des besoins et le contrat venaient d'être établis. TRUCCO© est une multinationale spécialisée dans la vente de vêtements.

Ce projet représente pour le client l'entrée dans le monde du e-Commerce, une démarche importante dans un marché en pleine expansion.

Suite à l'analyse, la plate-forme d'e-Commerce *open-source* MAGENTO a été choisie. Elle a été conçue spécialement pour le développement d'un site de vente en ligne et offre toutes les fonctionnalités nécessaires.

La première partie de mon stage fut dédiée à mettre en place une architecture physique capable d'offrir ce service en grand volume, avec un taux élevé de fiabilité et de sécurité.

La seconde partie du projet est consacrée à l'intégration entre SAP et MAGENTO, le client désirant intégrer entièrement la vente en ligne dans son système de gestion usuel.

Le projet a été établi en quatre phases :

- Mise en place de l'infrastructure hébergeant le site
- Intégration MAGENTO-SAP
- Intégration MAGENTO-CRM
- Mise en production

Ce travail relate l'implémentation, en trois mois, des deux premières phases et démontre sa viabilité d'une manière empirique.

Environnement de Travail

L'équipe en charge du projet était constituée de deux membres, un développeur spécialisé en RUBY et SQL et un chef de projet. Suite à une décision interne à Birchman, le chef de projet a été remplacé trois semaines après mon arrivée et un quatrième membre a rejoint l'équipe. Ce nouveau collègue a été intégré pour aider au développement du code de MAGENTO, principalement pour ses connaissances du ZEND Framework sur lequel Magento a été construit.

Le choix de Magento et des serveurs a été fait durant l'analyse avant mon entrée dans le projet. Ces choix ont eu plusieurs conséquences sur mon environnement de travail.

Les serveurs sont deux machines CentOS (Linux RHEL) et ceci a orienté le choix de PYTHON comme langage de développement pour les outils d'intégration.
MAGENTO quant à lui requiert une structure LAMP ainsi qu'une base de données MySQL.

Outils de développement :

Aptana – IDE variante d'ECLIPSE spécialisée pour le développement WEB et PYTHON.

Git et SVN – En interne Birchman utilise GIT pour le contrôle de versions. Pour ce qui est du contrôle de versions sur les serveurs de tests, SVN a été choisi comme outil.

VMware – Durant la développement il a été nécessaire d'avoir une installation de Magento

MySQL WorkBench – Pour ce qui est des consultations et des modifications à la base de données, cet outil a été fort utile. Si sur mon environnement virtuel PHP-MyAdmin était un bon outil, il était inacceptable de l'installer sur un serveur exposé au web à cause des nombreuses failles de sécurité qu'il contient. MySQL WorkBench a été un outil particulièrement facile d'utilisation une fois la sécurité mise en place, car il supporte la connexion à une base de données au travers d'une connexion SSH.

Vi – Au départ, je ne comprenais pas pourquoi ce programme, très ancien, était encore tant utilisé. J'ai néanmoins utilisé Vi tout au long de mon stage pour toutes mes modifications de fichiers sur le serveur et j'ai dû constater que Vi reste un outil d'édition surpuissant une fois que l'on s'est adapté à son interface et qu'il offre des fonctionnalités spectaculaires.

Serveurs, Sécurité et Réseaux

Elaboration d'une infrastructure robuste, optimisée et sécurisée sous CentOS

Description générale:

La première tâche qui m'a été confiée au sein de l'entreprise fut d'étudier et d'implémenter l'infrastructure nécessaire au bon fonctionnement de MAGENTO.

Les serveurs choisis par le client découlent d'un accord préexistant avec une compagnie de *hosting*. Ceux- ci ont donc mis à notre disposition deux serveurs, l'un étant destiné à devenir le serveur WEB et l'autre la base de données. Les serveurs sont deux VPS, chacun faisant tourner la dernière Edition de CentOS-6.2[2]. Ils possèdent entre eux une connexion LAN à haut débit ainsi qu'une connexion ADSL suffisante pour accommoder un trafic WEB important.

Quand le problème m'a été exposé, on m'a averti que MAGENTO souffrait de problèmes de performance et que mon objectif premier serait d'arriver à optimiser les serveurs pour obtenir une qualité de service qui répond aux attentes d'une plateforme grand publique.

Une fois les serveurs en production, le deuxième aspect primordial dans l'élaboration d'une plate-forme commerciale est la sécurité. Une faille dans la sécurité d'un site de vente en ligne peut se révéler catastrophique et avoir un impact négatif non seulement sur les bénéfices du projet mais aussi sur la réputation de la compagnie.

Ce projet a un second objectif, créer une architecture type pour les futurs projets MAGENTO de la compagnie. Par conséquent, il était extrêmement important de documenter et de justifier les choix d'implémentation. Tout au long de ce chapitre, démontre les résultats obtenus de manière à ce que lecteur puisse se faire une idée de l'impact des différents choix d'implémentation, tout en stipulant les avantages et les inconvénients de chaque technique.

Même si je ne présente ici qu'une seule configuration, le résultat final proposé en conclusion de ce stage est le fruit de plus d'un mois de tests et d'essais avec de multiples configurations différentes. Si il n'existe pas de fin aux possibilités d'optimisation, la solution apportée répond aux objectifs du projet tout en restant aisément déployable et stable.

---

[2] Wikipedia: *Virtual Private Server* - http://en.wikipedia.org/wiki/Virtual_private_server

Objectifs formels :

- Installer et configurer MAGENTO
- Optimiser les temps de réponse et l'utilisation de bande passante du serveur
- Implémenter la sécurité nécessaire pour le commerce en ligne
- Installer et configurer le contrôle des versions pour les différents acteurs
- Documenter la configuration

Le Matériel :

Ceci est la configuration des VPS fournis par le client, cette configuration a été choisie par le client pour effectuer son insertion sur le marché de l'e-Commerce. En tant que nouvelle démarche commerciale la configuration est d'un budget moyen mais offre la possibilité d'extension future si la demande devait s'en faire ressentir.

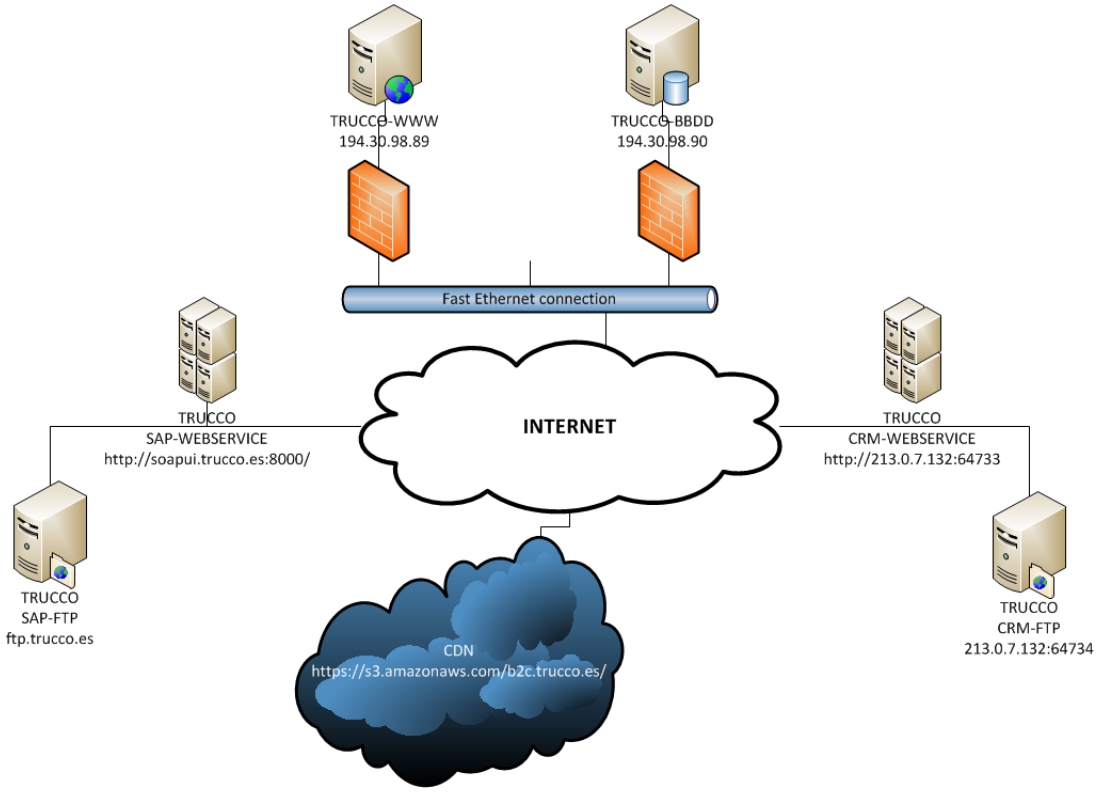
Serveur Web – TRUCCO-WWW:

- CPU : 8 cœurs
- RAM : 8GB
- OS : CentOS 6.2
- Disque dur : 200GB

Serveur de base de données – TRUCCO-BBDD:

- CPU : 6 cœurs
- RAM : 6GB
- OS : CentOS 6.2
- Disque dur : 100GB

Cette configuration doit être intégré avec les services existant de TRUCCO tel le CDN et les serveurs de web service qui seront utilisé dans des phases ultérieure du projet. Voici l'infrastructure complète du réseau :



Installation de MAGENTO:

Ici est détaillé une première installation fonctionnelle de MAGENTO. Dans la suite du document, ces configurations seront expliquées en détail et optimisées.

Modules requis :

Avant de commencer l'installation de MAGENTO, certains services doivent être obligatoirement installés. Outre une solution de type LAMP MAGENTO a certaines dépendances envers des modules PHP. Dans un souci de maintenance et de sécurité, l'entièreté des services installés provient de dépôts (*repository)*.

Un effort a été fait en ce sens pour permettre l'automatisation des mises à jour du système et ainsi augmenter la sécurité. De fait, la majorité des vulnérabilités d'un serveur viennent de programmes non mis à jour[3]. En assurant une méthode uniforme et sécurisée de mises à jour, ce problème peut être réduit.

[3] CCSEO : *The importance of updates... -http://www.ccseo.com/blog/704/the-importance-of-updates-backups-security-hardening/*

MAGENTO requiert les modules PHP suivants :

- PDO/MySQL
- MySQLi
- mcrypt
- mhash
- simplexml
- DOM

Certains de ceux-ci (MCRYPT et MHASH) ne sont pas disponibles dans le dépôt officiel de CentOS mais il existe plusieurs dépôts alternatifs contenant tout ce dont nous avons besoin. Choisir un dépôt implique une grande confiance dans la source : s'il est contaminé par des programmes non fiables, la sécurité ainsi que le fonctionnement du serveur pourraient être mis en danger.

Le choix s'est porté sur le dépôt de REMI qui est régulièrement mis à jour ainsi que conseillé dans la documentation officielle de CentOS[4]. Ceci permet d'utiliser les fonctionnalités standard de mise à jour de CentOS au travers de l'utilitaire YUM pour l'entièreté des services installé[5].

Emplacement de l'installation :

Le partitionnement des serveurs a été fait de la manière suivante :

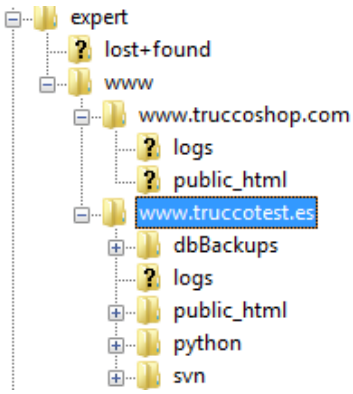
1. Une partition de 12GB dans laquelle est contenue l'entièreté du système d'exploitation.
2. Une partition physique additionel 'expert' de 200GB pour le contenu WEB.
3. Une partition Swap de 4GB.

Ce modèle de partitionnement n'est pas optimal, particulièrement au niveau de la sécurité mais il est imposé par la compagnie de hosting. SARENET fournit un partitionnement standard pour CentOS, celui-ci n'est pas modifiable car de cette structure dépendent les scripts automatisés de backups.

MAGENTO est donc installé sur la seconde partition physique, une arborescence est ainsi créée qui contiendra deux installations, l'une destinée à la production et l'autre au développement. Une solution de backups automatisé de qualité est offerte par SARENET et a été choisie par le client lors de l'analyse.

[4] Dépôt de REMI - http://rpms.famillecollet.com/
[5] Fedora Project : *YUM* - http://fedoraproject.org/wiki/Yum

Chacune de ces installations réside dans son propre dossier contenant l'entièreté des fichiers requis et chacune correspond à un VirtualHost différent.



Une première configuration d'Apache pour MAGENTO donne accès à ces installations de la manière suivante :

```
NameVirtualHost *:80

<VirtualHost *:80>
ServerAdmin maxime.frydman@birchmangroup.com
ServerName truccoshop.com
ServerAlias www.truccoshop.com
DocumentRoot /expert/www/www.truccoshop.com/public_html/
ErrorLog /expert/www/www.truccoshop.com/logs/error.log
CustomLog /expert/www/www.truccoshop.com/logs/access.log combined
</VirtualHost>

<VirtualHost *:80>
ServerAdmin maxime.frydman@birchmangroup.com
ServerName truccotest.es
ServerAlias www.truccotest.es
DocumentRoot /expert/www/www.truccotest.es/public_html/
ErrorLog /expert/www/www.truccotest.es/logs/error.log
CustomLog /expert/www/www.truccotest.es/logs/access.log combined
</VirtualHost>
```

Ceci apporte déjà la majorité des fonctionnalités nécessaires au bon fonctionnement des sites, particulièrement des *logs* séparés pour chaque installation permettant d'identifier séparément les erreurs de chacune. Le client a acheté le nom de domaine www.truccoshop.com et correspond à l'installation de production, le nom de domaine www.truccotest.es est uniquement utilisé en interne pour le développement.

La dernière étape consiste à extraire l'archive de MAGENTO dans les répertoires publiques de la configuration CAD: …/public_html.

Performance :

Le temps de réponse d'un site est primordial. Plusieurs études on démontré un lien de cause à effet entre le temps de chargement d'une page et la satisfaction des utilisateurs[6].

Chaque année le temps d'attente acceptable diminue : là où en 2000 les études de Akamai indiquaient que huit secondes était un temps moyen acceptable pour les utilisateurs, en 2006 la même étude a révélé que ce temps avait chuté à quatre secondes. La dernière étude, parue en 2009, indique que ce temps est maintenant tombé à deux secondes[7].
Quarante-sept pour cent des utilisateurs s'attendent à un temps de réponse en dessous des deux secondes, mais que se passe-t-il quand ce temps est dépassé ? Selon les mêmes études, dès que le temps d'attente dépasse les trois secondes, un site perd quarante pour cent de ses visiteurs, le temps moyen sur le site chute et le taux de conversion (visiteurs devenant acheteurs) diminue.
Il n'est donc pas surprenant que la performance soit un point primordial dans l'élaboration d'une plate-forme à succès.

L'objectif a été fixé à deux secondes. Le temps de réponse initial était de sept secondes pour l'index uniquement et avec un seul utilisateur connecté. Nous verrons par la suite qu'avec les bonnes optimisations, l'objectif est atteignable et que dans certaines conditions on peut le faire chuter en dessous d'une seconde.

Outils de mesure :

Les outils utilisés pour analyser les temps de réponse sont les suivants :

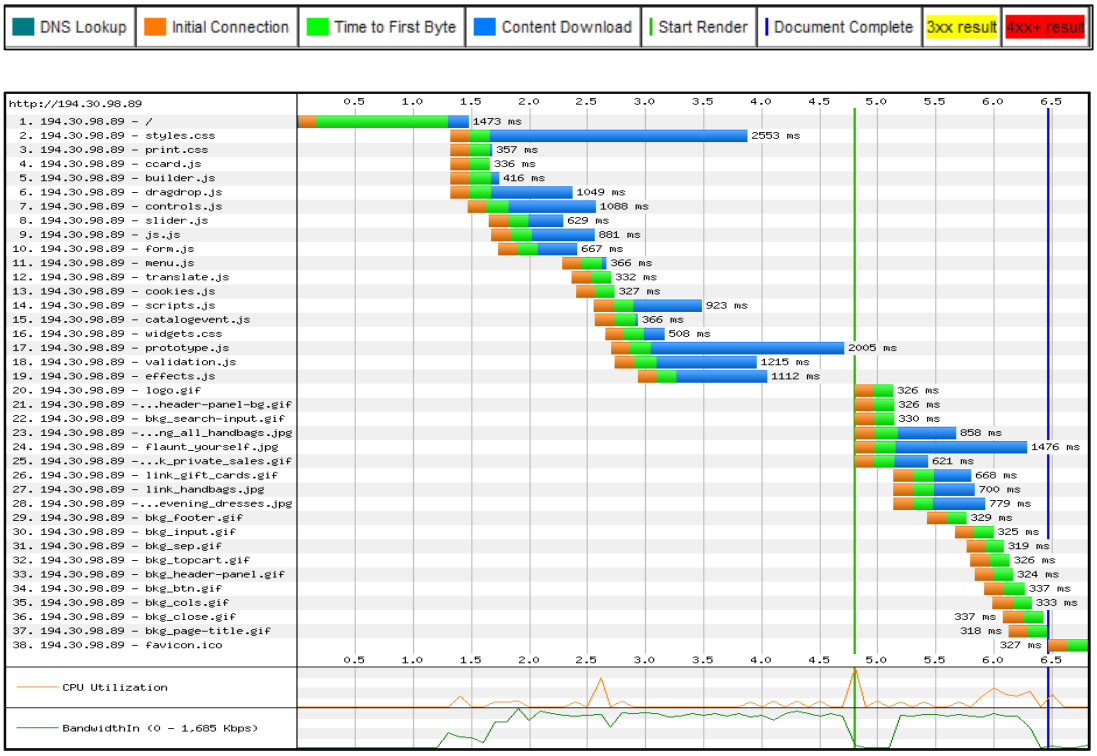
- WEBPagetest - http://webpagetest.org
- Apache Benchmark – http://httpd.apache.org/docs/2.0/programs/ab.html
- Siege - http://joedog.org/siege-home/

[6] Nah, F.: *A Study on Tolerable Waiting Time* - http://cba.unl.edu/people/fnah/
[7] Akaimai : *2 seconds rule* :
http://www.akamai.com/html/about/press/releases/2009/press_091409.html

WEBPagetest :

WEBPagetest a été développé pour usage interne par AOL et est distribué sous licence BSD depuis 2008. Cet outil permet d'analyser les temps de réponse de page WEB mais offre aussi des informations complémentaires permettant de déterminer la cause de chaque ralentissement.
Les tests sont effectués avec Internet Explorer comme navigateur de référence, représentant trente-cinq pour-cent de parts de marché: IE reste le navigateur le plus utilisé actuellement[8].

Une première analyse du site donne les résultats suivants :



Ceci représente le temps entre le moment où la requête est faite, le début de l'affichage de la page (barre verticale verte) et l'affichage complet (barre verticale bleu). L'axe horizontal représente le temps en secondes et l'axe vertical les différents éléments présents sur la page.

[8] Wikipedia: *Usage share of web browsers* - http://en.wikipedia.org/wiki/Usage_share_of_web_browsers

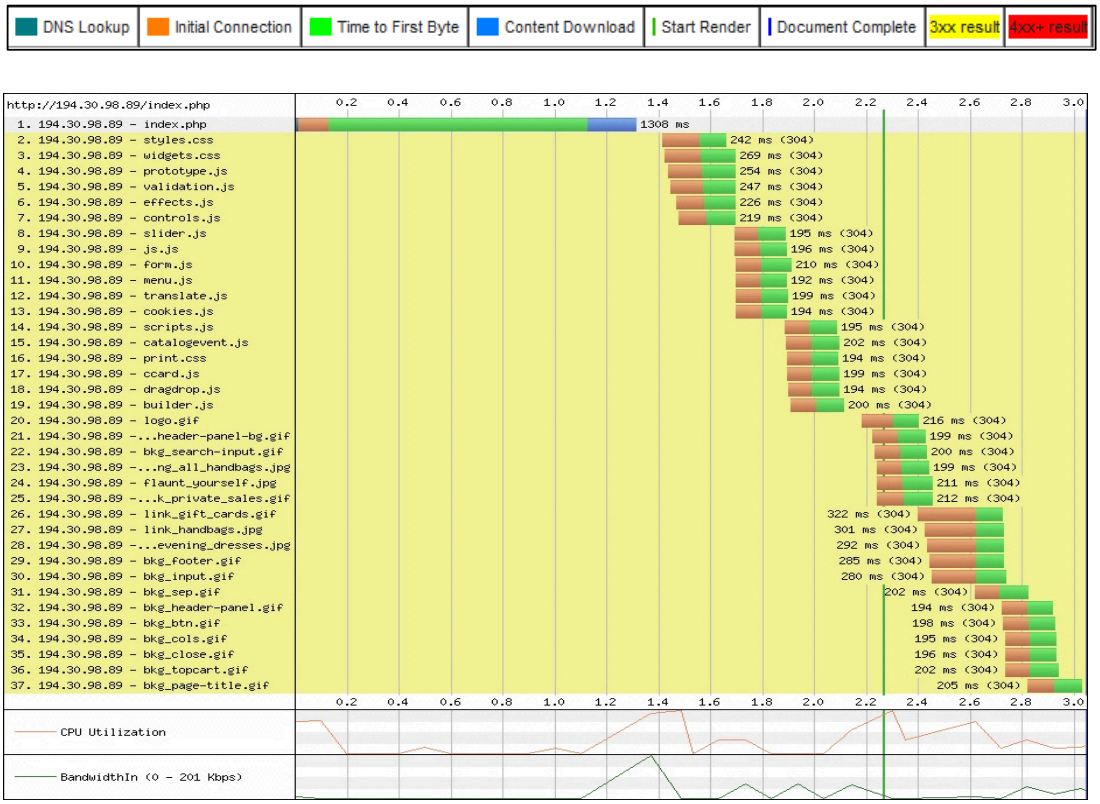
Il y a plusieurs informations intéressantes à retenir de ce graphique :

- Chaque élément est précédé par un temps de connexion (en orange) ; celui-ci indique le temps nécessaire pour établir une connexion TCP avec le serveur.

- Le *time to first byte (en vert)* indique le temps de calcul nécessaire au serveur avant de répondre à une requête.

- *Content download (en bleu)* qui indique le temps nécessaire à l'envoi des informations du serveur au client.

La seconde information produite par WEBPagetest est le temps nécessaire aux visites consécutives. Tous les navigateurs modernes implémentent une forme ou l'autre de *mise en cache* qui permet de réduire les données qui doivent être transmises.



Similairement au premier graphique nous repérons les trois indicateurs: on remarque qu'ici le temps de téléchargement du contenu a presque disparu. Il est indiqué par le code de réponse HTTP 304 ou *not modified* qui montre la technologie de *mise en cache* en action[9].

Siege et Apache Benchmark

Siege et *Apache Benchmark* (*AB*), servent à tester les capacités de concurrence du serveur web. Là où WEBPagetest produit des données très précises sur chaque élément contenu dans une page, *Siege* et *AB* eux ne s'intéressent qu'au temps de réponse du serveur. Ce temps de réponse est équivalant au temps nécessaire pour générer la page dans son entièreté (exécution du code, requête à la base de données…) sans prendre en compte le temps nécessaire au téléchargement des éléments.

Les deux outils sont fort similaires. *Siege* permet de simuler un trafic « humain » en testant de multiples pages et en induisant des délais entre les différentes requêtes, là où *AB* se concentre sur le test d'une seule page sans aucun répit.

Les deux outils permettent d'automatiser les tests en spécifiant différents paramètres, le principal étant le nombre de connexions simultanées à simuler et le format de représentation des résultats. Le format utilisé pour récolter les données fut le CSV, ceci a permis d'utiliser des outils tel qu'Excel pour estimer et comparer entre elle les performances des différentes configurations testées.

[9] W3: *Hypertext transfer protocol codes* - http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Un premier test du serveur avant optimisation avec dix connexions simultanées montre déjà des temps de réponse élevés et bien au delà des 2 secondes visées.

```
$ ab -c 10 -n 500 -A BirchAdmin:Birchman2012 -g 5.tsv http://194.30.98.89/index.php
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 194.30.98.89 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Finished 500 requests


Server Software:        Apache/2.2.15
Server Hostname:        194.30.98.89
Server Port:            80

Document Path:          /index.php
Document Length:        20062 bytes

Concurrency Level:      10
Time taken for tests:   122.523 seconds
Complete requests:      500
Failed requests:        0
Write errors:           0
Total transferred:      10260500 bytes
HTML transferred:       10031000 bytes
Requests per second:    4.08 [#/sec] (mean)
Time per request:       2450.458 [ms] (mean)
Time per request:       245.046 [ms] (mean, across all concurrent requests)
Transfer rate:          81.78 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       43   71 267.2     47    3045
Processing:  1556 2368 178.3   2382    3126
Waiting:     1467 2238 174.8   2249    2999
Total:       1601 2439 300.5   2435    5428

Percentage of the requests served within a certain time (ms)
  50%   2435
  66%   2496
  75%   2529
  80%   2552
  90%   2624
  95%   2705
  98%   2789
  99%   3179
 100%   5428 (longest request)
```

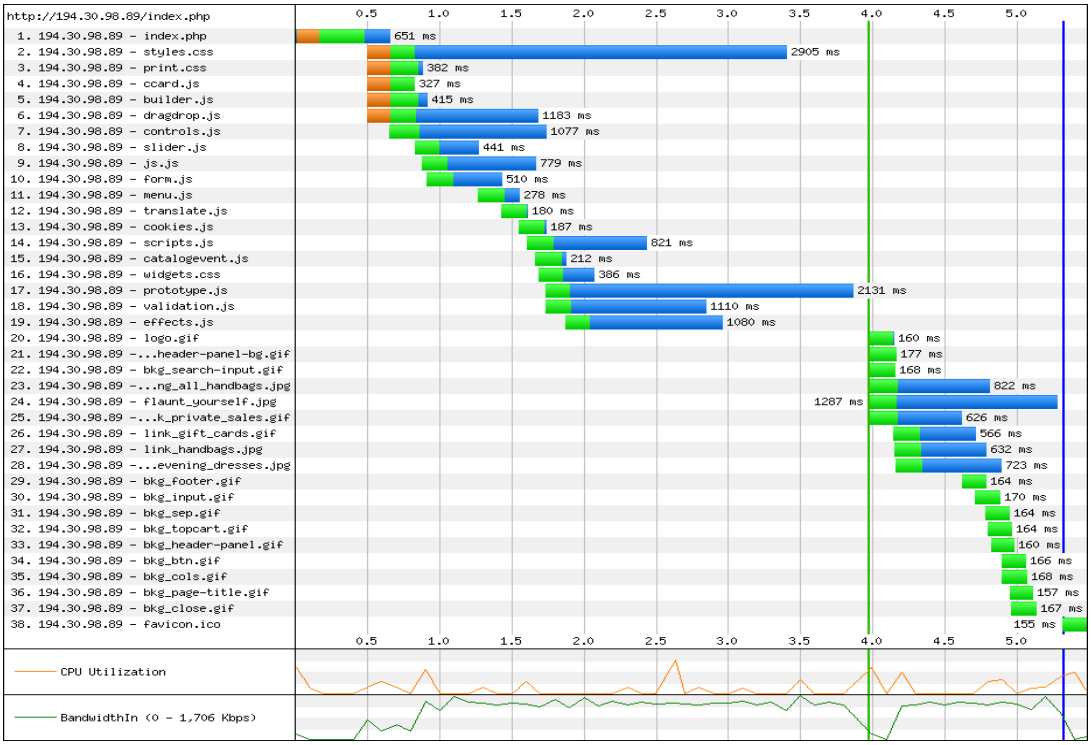
TCP et *keep-alive*

La première piste à explorer pour optimiser le temps de réponse est le *keep-alive*. Il permet de maintenir la connexion TCP ouverte pour de multiples requêtes http provenant du même

utilisateur. Ainsi, en évitant de générer une nouvelle connexion pour chaque éléments de la page, on élimine ce temps de latence. Le *keep-alive* se définit dans le fichier de configuration d'Apache avec les trois directives suivantes:

```
Keep-alive on
MaxKeep-aliveRequests 100
Keep-aliveTimeout 15
```

La première active le *keep-alive*, la seconde définit le nombre maximum de requêtes pouvant être traitées par une connexion et la troisième, le temps d'attente maximale entre deux requêtes (en secondes). Sans aucune autre modification, l'on remarque immédiatement un gain de près d'une seconde au temps de chargement, ce qui nous amène a 5.5 secondes, soit un gain de ± quinze pour cent.



Le bénéfice de cette optimisation varie selon les pages parce qu'il dépend du nombre d'éléments pour lesquels une requête doit être émise.

Outre la vitesse de réponse, le *keep-alive* offre un deuxième avantage : une réduction de la charge sur le processeur qui n'a pas à établir une nouvelle connexion par élément. En contrepartie on augmente l'utilisation de mémoire vive, car il faut stocker les paramètres de connexion pendant une plus longue durée.

Pour la majorité des serveurs, la mémoire vive disponible est un point sensible. Dans le cas de MAGENTO nous verrons plus loin qu'il existe d'autres limitations qui apparaissent avant que la mémoire ne vienne à manquer.

Mod_Deflate et Mod_Expire

La seconde piste à explorer est le temps de téléchargement.

Il y a existe deux approches pour réduire le temps d'envoi :

1. Augmenter la bande passante
2. Réduire la taille des données

En considérant que nous ne pouvons pas à ce stade modifier la bande passante du serveur, nous allons nous concentrer sur des technologies permettant de compresser les données ainsi qu'aider les navigateurs à mettre les éléments en cache.

Mod_Deflate est un module Apache permettant de compresser le contenu à la volée avant qu'il ne soit envoyé au client[10]. Ceci a pour avantage de réduire significativement la taille des pages mais peut poser des problèmes en termes d'utilisation de processeur, s'il n'est pas configuré prudemment.

L'avantage d'une compression à la volée se fait immédiatement ressentir. Dans notre cas, cela représente un gain de septante-huit pour-cent sur la taille des fichiers statique de l'index (HTML, CSS, JavaScript).

Il serait tentant de tout comprimer de cette manière, particulièrement les contenus lourds tels que les images, malheureusement la compression a un impact élevé sur l'utilisation du processeur et n'est pas applicable pour ce type de contenu[11]. Mod_Deflate utilise GZip comme algorithme de compression. Celui-ci est extrêmement efficace pour des fichiers texte de petite taille mais le temps de compression pour les fichiers plus lourds réduit les avantages de la compression.

La configuration de Mod_Deflate peut se faire au niveau d'Apache de la manière suivante :

```
<ifmodule mod_deflate.c>
        SetOutputFilter DEFLATE
        #Problèmes avec Netscape 4.x
        BrowserMatch ^Mozilla/4 gzip-only-text/html
        #Problèmes avec Netscape 4.06-4.08
```

[10] Apache: *Module mod_deflate* - http://httpd.apache.org/docs/2.0/mod/mod_deflate.html
[11] Web Performance: *Measuring performance effects of mod_deflate* - http://webperformance.com/library/reports/moddeflate/

```
        BrowserMatch ^Mozilla/4\.0[678] no-gzip
        #Reperer IE qui pretend etre Netscape
        BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
         # Ne pas compresser les images
        SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png)$ no-gzip dont-vary
        # Prendre en compte certain proxy
        Header append Vary User-Agent env=!dont-vary
</ifmodule>
```

Outre quelques règles spécifiques à certaines versions de navigateurs défectueux et au *proxy*, il est intéressant de noter l'expression régulière qui définit ce qui ne doit pas être compressé. Il y a deux approches possibles, le *white listing* et le *black listing* dans ce cas-ci on utilise un *black listing* interdisant les images.

Mod_Expire quant à lui est une optimisation au niveau du client. Les navigateurs modernes utilisent tous un système de cache permettant d'identifier si du contenu téléchargé préalablement peut être réutilisé[12].

Comme nous l'avons vu dans le diagramme de visites consécutives, par défaut, quand le navigateur repère du contenu déjà téléchargé, fera une requête au serveur pour savoir si l'information est toujours à jour.

Ceci entraine un va et vient de requêtes coûteux et inutile. Dans l'exemple on remarque près de 1.6 secondes de communication pour se rendre compte qu'il n'y a rien à faire. En précisant manuellement le temps de fraicheur des informations on donne explicitement l'information aux navigateurs qui, dès lors, n'ont plus besoin de la confirmation du serveur.

La question à se poser est si ce comportement est correct pour l'entièreté du contenu du site. Mal configuré, on risque de se retrouver dans une situation où des changements de contenu ne seront pas reflétés au niveau de l'utilisateur tant que sa cache n'est pas nettoyée.

Pour un site de vente tel que celui que nous construisons ceci ne représente pas un réel problème. Les pages de contenu dynamique en PHP eux ne sont jamais mis en cache car leur contenu est généré dynamiquement par le serveur. Ainsi, ce qui est stocké en cache sont les ressources statiques référencées dans chaque page HTML tel le code JavaScript, les CSS ainsi que les images. Ce contenu, une fois mis en place, a tendance à ne jamais être peu modifié sauf à l'occasion d'une mise à jour du Framework.

Vu le fonctionnement de MAGENTO et l'utilisation d'un CDN, la configuration au niveau du site est extrêmement simple et est unique au niveau du site entier. Pour une configuration

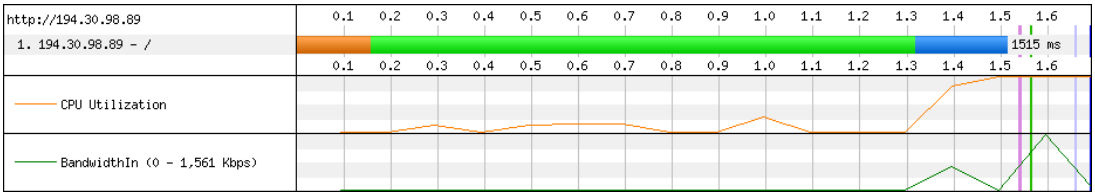
[12] Apache: *Module mod_expire* - http://httpd.apache.org/docs/2.0/mod/mod_expires.html

nécessitant plus de précision on peut tirer profit du fonctionnement d'Apache en définissant des configurations spécifiques à chaque dossier au travers des fichiers « .htaccess ».

.htaccess à la racine du site :

```
<IfModule mod_expires.c>
        ExpiresActive on
        ExpiresDefault "access plus 1 month"
        ExpiresByType image/gif "access plus 7 days"
</IfModule>
```

Le résultat est exactement ce que nous pouvions espérer. Une fois le contenu mis en cache, le navigateur est entièrement capable de générer la page sans aucun contenu additionnel. En comparaison avec les premiers on remarque un gain de 1.5 secondes (1.5^{s} pour charger l'index en comparaison à 3^{s}).



Ceci est effectivement le résultat attendu mais n'affecte que les vues consécutives de contenu, le temps d'affichage au premier chargement est inchangé. Mais une grande partie du contenu statique (CSS, JavaScript, logos) est partagé entre les différentes pages du site, on profitera donc de l'optimisation pour la majorité des pages, une fois le premier accès chargé.

PHP-FPM, PHP-FCGI, HTTP-MPM et la cache APC

L'ultime technique d'optimisation que nous allons explorer est liée au temps d'exécution du contenu dynamique. Ce temps peut être identifié *(en **vert**)* sur les graphiques pour les pages PHP. Cela correspond au temps entre la réception d'une requête du client et le moment où la page a été entièrement générée et est prête à être envoyée au client.

Il y a plusieurs facteurs qui entrent en jeu. Voyons la séquence d'étapes nécessaires pour émettre une réponse avec la configuration de base d'Apache et PHP.

- A la réception d'une requête le démon http crée un nouveau processus pour ce client.
- Ce processus lance une nouvelle instance de PHP
- Le code est interprété et exécuté

Une première amélioration est d'éviter la création d'un nouveau processus pour chaque requête http. La création d'un nouveau processus est coûteuse en termes de mémoire, car un contexte entier est nécessaire pour chacun, ainsi qu'en termes de latence. Le Multi-Processing Module - Worker d'Apache (MPM) offre la possibilité d'avoir un nombre limité de processus chacun traitant de multiples requêtes en parallèle[13]. De plus, il est possible de configurer MPM de manière à ce qu'il pré-alloue de nouveaux processus avant que ceux existant arrivent à leur limite de capacité. De cette manière, le processus est déjà instancié avant la requête du client et le temps d'instanciation ne l'affecte pas.

Ce partage de ressources a un désavantage. Si un crash devait subvenir dans une des requêtes, c'est l'entièreté des clients qui partagent le processus qui en paieront le prix. Des crashs récurrents du processus Apache seraient un indicateur de problème dans le code exécuté. En considérant que nous utilisons une plate-forme stable et de qualité, il est peu probable que cela pose problème.

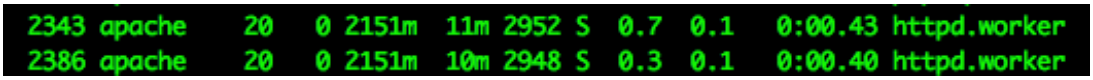
Un test de concurrence démontre les gains de mémoire que nous apporte MPM.

```
top - 13:11:57 up 1 day,  8:31,  1 user,  load average: 1.04, 0.75, 0.31
Tasks: 139 total,  11 running, 128 sleeping,   0 stopped,   0 zombie
Cpu(s): 95.3%us,  4.3%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
Mem:   8028984k total,  1566604k used,  6462380k free,    72364k buffers
Swap:  2097144k total,        0k used,  2097144k free,   317560k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
30309 apache    20   0  345m  50m 4408 R 29.9  0.6   0:38.95 httpd
30263 apache    20   0  440m 145m 4840 R 26.3  1.9   0:43.74 httpd
30279 apache    20   0  345m  51m 4416 S 24.9  0.7   0:40.38 httpd
30292 apache    20   0  344m  50m 4416 R 23.6  0.6   0:38.72 httpd
28433 apache    20   0  345m  50m 4428 S 22.6  0.7   1:26.04 httpd
28154 apache    20   0  345m  51m 4436 S 21.6  0.7   1:30.62 httpd
28398 apache    20   0  345m  51m 4428 R 21.6  0.7   1:24.06 httpd
30307 apache    20   0  345m  51m 4428 S 21.6  0.7   0:38.65 httpd
28413 apache    20   0  345m  51m 4428 S 21.3  0.7   1:23.69 httpd
30297 apache    20   0  350m  56m 4556 S 20.9  0.7   0:39.41 httpd
30143 apache    20   0  345m  51m 4412 R 20.6  0.7   1:01.47 httpd
30273 apache    20   0  345m  51m 4408 S 19.9  0.7   0:38.81 httpd
30252 apache    20   0  345m  51m 4412 R 18.6  0.7   0:39.97 httpd
30284 apache    20   0  345m  51m 4456 R 18.6  0.7   0:39.30 httpd
30253 apache    20   0  345m  51m 4428 S 17.9  0.7   0:40.41 httpd
30302 apache    20   0  345m  51m 4420 R 17.3  0.7   0:39.29 httpd
30255 apache    20   0  345m  51m 4436 R 14.0  0.7   0:38.57 httpd
28397 apache    20   0  347m  52m 5536 R 13.6  0.7   1:27.00 httpd
28406 apache    20   0  345m  51m 4432 S 13.3  0.7   1:24.53 httpd
30299 apache    20   0  345m  51m 4408 S 11.3  0.7   0:39.00 httpd
```

Avec dix connections concurrentes on remarque le grand nombre de processus, chacun consommant ± 70Mb de mémoire vive. De plus, le *keep-alive* que nous avons activé précédemment prolonge la vie de chacun de ces processus de quinze secondes.

[13] Apache: *MPM Worker* - http://httpd.apache.org/docs/2.2/en/mod/worker.html

Avec MPM activé, on remarque deux processus, consommant une fraction de la mémoire.



La configuration MPM:

```
<IfModule worker.c>
        StartServers          4
        MaxClients          300
        MinSpareThreads      25
        MaxSpareThreads      75
        ThreadsPerChild      25
        MaxRequestsPerChild   0
</IfModule>
```

Ceci est une configuration relativement standard pour MPM. Il peut être intéressant de jouer avec le nombre de clients maximal (ici 300), cela permet d'établir une limite au nombre de requêtes concurrentes qui sont traitées. Si cette valeur est trop basse, les ressources du serveur seront sous-utilisées car les connexions, au-delà de cette limite, sont mises en file d'attente. Si elle est trop haute, on saturera les ressources et on perdra tous les bénéfices des optimisations. Une formule est conseillée pour le calcul de cette valeur :

```
MaxClients ≈ (RAM - taille_des_autres_processus)/(taille_processus_apache)
```

Ensuite il convient d'effectuer des tests pour s'assurer de rester dans une utilisation de mémoire élevée mais ne dépassant pas les ressources disponibles[14].

Nous avons donc réussi à limiter le nombre de processus Apache du serveur mais cela ne suffit pas. En effet, le site est composé entièrement de pages PHP. Si on a regroupé les requêtes http en un petit nombre de processus multithread, chacun de ces threads instancie son propre processus CGI pour l'interprétation du contenu dynamique.

Entre PHP-FPM ou PHP Fast Process Manager, similairement au MPM celui ci permet de configurer un petit nombre dynamique de processus qui traiteront les requêtes CGI dans un environnement multithread[15].

Un des grands avantages de cette technique est que toutes les requêtes passent par un seul processus qui les aiguille vers un de ses sous processus d'interprétation. Ce niveau de

[14] Developer Side : *Apache performance tuning* - http://devside.net/articles/apache-performance-tuning
[15] PHP-FPM - http://php-fpm.org

contrôle supplémentaire amène des fonctionnalités intéressantes telles des files d'attente quand le nombre maximal de requêtes concurrentes est atteint. Si on configure le nombre de processus d'interprétation en tenant compte des ressources du serveur, ceci permet d'éviter l'écroulement du système en cas de surcharge.

Le nombre de processus maximal d'interprétation se calcule en fonction de la mémoire vive et de la taille de chaque processus d'interprétation similairement au MPM. Un second paramètre intéressant est la durée de vie de chaque interpréteur. Comme cette configuration amène à des processus persistants et non plus dépendant des requêtes, ceci amène un problème de possible fuite de mémoire particulièrement pour un langage comme PHP dépendant d'un garbage collector. Pour éviter cela, nous allons indiquer le nombre de requêtes maximales que peut traiter un interpréteur avant d'être recyclé. Cette valeur doit être choisie avec attention : si elle est trop petite on perd l'avantage des processus persistants et si elle trop grande on risque de saturer la mémoire vive avec les fuites. En général cela amène à une durée de vie de 200-300 requêtes par processus, mais il faut prendre en compte que nous avons activé le *KeepAlive* et ceci a pour conséquence que plusieurs requêtes du même utilisateur seront traitées comme une seule par le FPM. Ceci nous amène à une valeur plus conservative de l'ordre de 50 requêtes par interpréteur.

```
[www]

listen = /var/run/php-fpm.sock

listen.allowed_clients = 127.0.0.1

listen.owner = apache
listen.group = apache
listen.mode = 0666

user = apache

group = apache

pm = dynamic

pm.max_children = 50

pm.start_servers = 5

pm.min_spare_servers = 5

pm.max_spare_servers = 35

request_slowlog_timeout = 20

slowlog = /expert/www/www.truccotest.es/logs/www-slow.log
php_admin_value[error_log] = /var/log/php-fpm/www-error.log
php_admin_flag[log_errors] = on

pm.max_requests = 50
```

Comme on le voit dans la configuration, *PHP-FPM* écoute sur un socket (/var/run/php-fpm.sock), c'est dans ce socket que doivent être envoyées les requêtes CGI.

Pour automatiser cette redirection nous allons faire appel à un autre module *Apache – mod_fastcgi*[16]. Celui-ci permet de redéfinir l'action à entreprendre pour chaque requête CGI et de la rediriger vers le socket de *PHP-FPM*. *Mod_fastcgi* est conçu pour ce type de redirection et nous permet de la configurer facilement au travers de la directive FastCGIExternalServer que nous faisons pointer vers le socket de *PHP-FPM.*

```
LoadModule fastcgi_module modules/mod_fastcgi.so

<IfModule mod_fastcgi.c>
    FastCGIExternalServer /usr/sbin/php-fpm -socket /var/run/php-fpm.sock -
idle-timeout 3600
    AddHandler php-fastcgi .php

    Action php-fastcgi /usr/sbin/php-fpm.fcgi
    ScriptAlias /usr/sbin/php-fpm.fcgi /usr/sbin/php-fpm
        <Directory /usr/sbin>
        Options ExecCGI FollowSymLinks
        SetHandler fastcgi-script
        Order allow,deny
        Allow from all
    </Directory>
</IfModule>
```

Il existe d'autres méthodes d'optimisation des processus CGI mais celle-ci a retenu mon attention, car elle est compatible avec l'APC que nous verrons par la suite. Il existe d'autres implémentations de *fastCGI* plus récentes ainsi que plus performantes que celle présentée ici, sauf que la cache APC ne peut pas être partagée par de multiples processus CGI indépendants. Le processus unificateur de *PHP-FPM* va nous permettre d'utiliser la cache APC sans restriction tout en jouissant des avantages de fastCGI.

La dernière optimisation que nous allons apporter est l'Alternative PHP Cache ou APC[17]. PHP est un langage interprété. Les langages interprétés sont plus lents que ceux qui sont précompilés. APC permet de stocker le travail d'interprétation en *opcode* et ainsi accélérer le temps d'exécution par la suite. On estime que cette optimisation réduite en moyenne le temps d'exécution d'un facteur de 2-10x.

De plus, MAGENTO peut être configuré pour stocker ses fichiers de configuration dans la cache APC, permettant ainsi de faire double emploi de la cache et de limiter les accès sur fichiers.

---

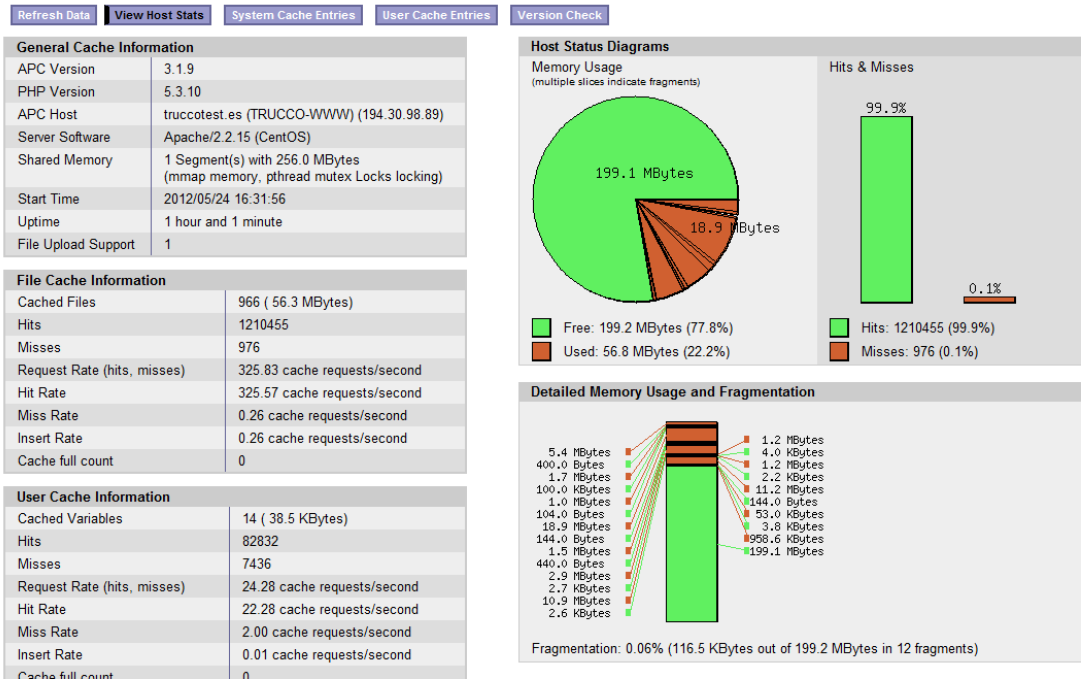
[16] Apache: *Module mod_fastcgi* - http://fastcgi.com/mod_fastcgi/docs/mod_fastcgi.html
[17] Drupal : *Alternative PHP Cache* - http://drupal.org/project/apc

Extrêmement simple à activer, *APC* offre fort peu de configuration, son mécanisme de cache est simple et efficace, le paramètre le plus important étant l'espace que l'on désire lui allouer. Dans une configuration standard, on considère 64-128MB comme plus que suffisant pour stocker les scripts en *opcode* mais, comme nous allons en plus y stocker les fichiers de configurations de MAGENTO, il est important d'augmenter cette valeur vers les 256MB.

Quand la cache *APC* est saturée, elle est automatiquement vidée et on redémarre à zéro, perdant ainsi tout ces bénéfices. Il vaut donc mieux allouer légèrement trop d'espace à l'*APC* plutôt que de se retrouver avec une cache constamment réinitialisée.

```
extension=apc.so
        [APC]
        apc.enabled = 1
        apc.optimization  = 0
        apc.shm_segments = 1
        apc.shm_size = 256M
        apc.ttl = 7200
        apc.user_ttl  = 7200
        apc.num_files_hint = 10000
        apc.user_entries_hint=10000
        apc.mmap_file_mask = /tmp/apc.XXXXXX
        apc.enable_cli = 1
        apc.cache_by_default  = 1
        apc.max_file_size = 5M
        apc.stat = 0
        apc.include_once_override = 1
```

Pour surveiller son bon fonctionnement la cache APC est fournie avec un utilitaire de gestion qui permet de voir exactement ce qu'elle contient. L'information la plus importante à surveiller est le taux de réussite (hits). Chaque foi qu'une ressource est demandée et qu'elle se trouve en cache, cela veut dire que nous avons joui d'un gain de performance.



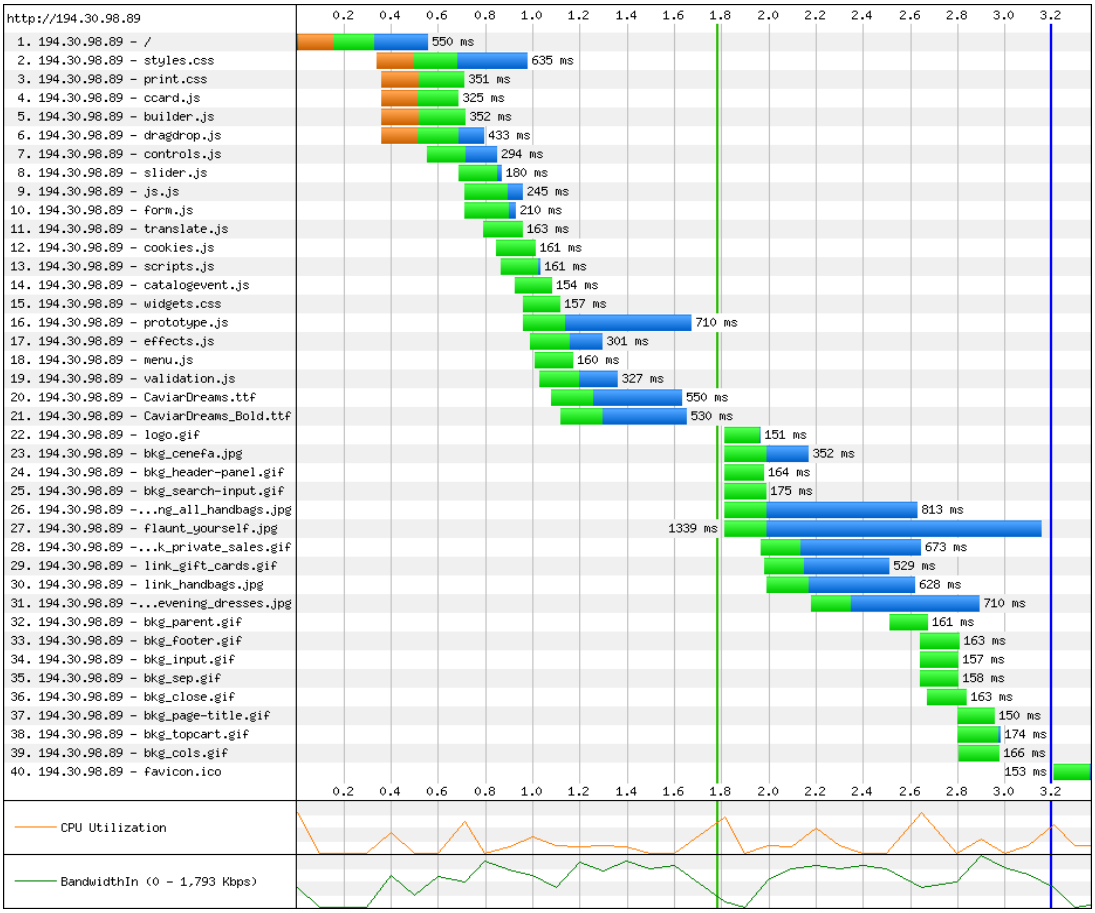
Comme nous le voyons ici, après un certain temps d'exécution, ce taux atteint près des 100%, ce qui est la valeur idéale et indique que tout fonctionne correctement.

La cache APC par contre est contraignante durant le développement. A chaque modification du code, si elle n'est pas nettoyée, les changements ne seront pas pris en compte. Il est donc conseillé de ne l'activer qu'au passage en production.

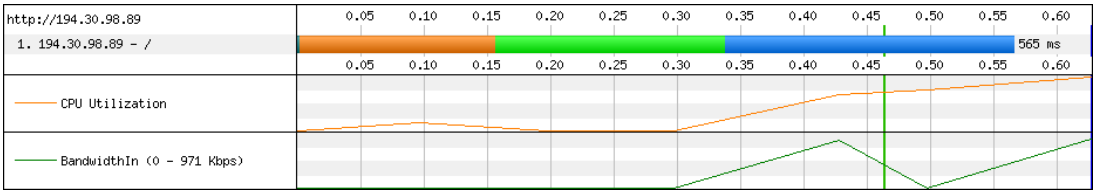
Ceci achève les optimisations de performance, la différence entre les tests initiaux et le résultat final est conséquent. Le temps de chargement a la première connexion au site a été réduit de plus moitié, ceci nous amène a 3.2 secondes pour la page d'index. Si ceci reste au dessus de nos objectifs mais il faut noter que ce temps ne s'applique que au chargement de la première page du site. Le début du rendu de la page commence a 1.8 seconde qui nous permet d'atteindre notre objectif. Le temps supplémentaire nécessaire au téléchargement du contenu ce verra réduit une fois le CDN mis en production, celui-ci malheureusement n'était disponible durant mon stage, mais il n'y a aucun doute qu'il apportera les ultimes améliorations nécessaire au site tant au niveau de la performance qu'au niveau de la scalabilité.

Le temps de réaction du site passé la première connexion lui a chuté a 0.6 seconde, ceci donne un aspect extrêmement réactif a la navigation et produit une expérience agréable aux visiteurs.

Analyse de première connexion :



Analyse de vue successive :



L'autre impacte a été d'augmenter la capacité du serveur en terme de concurrence. Initialement vingt connexion concurrente était suffisante pour ressentir un impact important sur la navigation suite a l'optimisation on est capable de gérer un maximum de 300 requêtes avant que ces mêmes effets ne se fassent ressentir. Ceci permet d'accommoder un trafique important au lancement de la plateforme mais il sera nécessaire de réévaluer la configuration après les premières semaine de mise en production selon la popularité du projet.

Sécurité :

Dans un projet d'E-Commerce la sécurité est un point crucial, cette année nous avons vu de multiples exemples des conséquences de piratage sur des sites et plateformes commerciaux[18]. L'impact des failles de sécurité sur les revenus des entreprises augmente annuellement et les compagnies réagissent en débloquant des fonds et du temps de développement supplémentaire pour sécuriser leurs projets.

Dans cette section je présente un ensemble de techniques et de bonne pratique pour protéger le site d'attaques éventuelles.

Pare-Feu :

Un pare-feu bien configuré est un bon point de départ dans l'élaboration d'une police de sécurité. Une fois que l'ensemble des services nécessaires au fonctionnement du serveur est établi, il s'agit de restreindre le trafic autorisé au strict minimum nécessaire a ces opérations.

Le serveur WEB offre deux services distincts, le premier est le contenu qu'il délivre aux visiteurs (http et https) et le second l'accès de gestion nécessaire à la maintenance du serveur (SSH). A cela s'ajoutent deux services de gestion imposés par l'hébergeur pour la maintenance et la surveillance de son parc informatique (QPIDD et GANGLIA).

CentOS est fourni avec un service de pare-feu IPTABLES[19]. IPTABLES permet la configuration des tables de pare-feu du *Kernel* Linux d'une manière simple mais efficace. Il existe deux façons de travailler avec ces tables, le *Whitelist* ou le *Blacklist*.

En mode *Whitelist,* on configure ce que l'on désire rejeter et ce qui n'est pas indiqué est autorisé; en mode *Blacklist*, c'est le contraire: on indique ce que l'on autorise et le reste est refusé.

Si chacune de ces méthodes a des avantages, dans le cas d'un serveur offrant des services bien précis, le *Blacklist* est l'unique solution réellement viable. Cette méthode est beaucoup plus restrictive et demande une bonne compréhension des besoins de chaque application mais permet de s'assurer qu'on n'expose aucun service non souhaité par mégarde.

---

[18] Network World: *Playstation Network hack will cost Sony $170M* - http://networkworld.com/news/2011/052311-playstation-network-hack-will-cost.html

[19] Netfiler: *Iptables* - http://netfilter.org/projects/iptables/index.html

Voici la configuration finale du serveur WEB:

```
#Authorizer les connections déjà établie
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
#Authorizer le trafique http and https
-A INPUT -p tcp --dport 80 -j ACCEPT
-A INPUT -p tcp --dport 443 -j ACCEPT
#Authorizer les connections ssh
-A INPUT -p tcp --dport 22 -j ACCEPT
#Authorizer qpidd et ganglia
-A INPUT -p tcp --dport 5672 -j ACCEPT
-A INPUT -p tcp --dport 8649 -j ACCEPT
-A INPUT -p udp --dport 8649 -j ACCEPT
#Authorizer le icmp et igmp (necessaire pour Ganglia et qpidd)
-A INPUT -p igmp -j ACCEPT
-A INPUT -p icmp -j ACCEPT
#Drop de tout le reste sauf les connections sortante
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT ACCEPT
```

sur des sites commerciaux

Il aurait été souhaitable de limité l'accès au service de QPIDD et GANGLIA ainsi qu'au connexion ICMP et IGMP au seuls IP de la compagnie de hosting mais celui-ci ne m'a pas fournit les informations nécessaire a l'implémentation.

Le serveur de base de données a une configuration similaire, si ce n'est qu'il offre uniquement le service de base de données. Une attaque courante est de *'bruteforcer'* les mots de passe des utilisateurs de la base de données. S'il existe des méthodes pour mitiger l'impact de ces attaques, la plus efficace est d'interdire toute connexion qui ne provient pas de notre serveur WEB.

```
-A INPUT -s 194.30.98.89/32 -p tcp -m tcp --dport 3306 -m state --state
NEW,ESTABLISHED -j ACCEPT
```

La première leçon apprise à mes dépens a été d'ajouter un script qui rétablit ces règles automatiquement à intervalle régulier. Iptables ne fait pas les choses à moitié: tout trafic qui n'est pas précisé dans les tables est sommairement éliminé. Suite à une erreur de manipulation de ma part, les tables se sont retrouvées vides; le résultat est un serveur qui n'accepte plus aucun trafic entrant et n'offre donc aucune possibilité de corriger la situation sans accès physique.

Il est important de noter qu'il vaut mieux placer le trafic le plus courant en tête de ces configurations, car pour chaque paquet entrant, la table est parcourue jusqu'à ce qu'une entrée lui correspondant est rencontrée. Si ceci a un impact infime sur la performance au

niveau d'un paquet, il ne faut pas oublier que le serveur est amené à traiter des milliers de paquets par seconde. On notera donc la première entrée de la configuration qui autorise toutes les connexions établies préalablement sans avoir à parcourir la table.

Sécurisation du protocole SSH:

Une recommandation typique quand on utilise SSH est de changer le port par défaut (**22**) pour éviter les tentatives de *brute forcing*. Dans notre cas, ceci n'est pas possible à cause de l'hébergeur et de ses outils de gestion qui requièrent un accès sur le port 22.

De ce fait, on est garanti de subir quotidiennement des milliers de tentatives de connexion SSH issues de botnets espérant pénétrer le système et en prendre le contrôle.

La première mesure de protection est de désactiver l'accès en tant qu'administrateur (ROOT), ce qui est l'attaque la plus courante et aux résultats les plus catastrophiques. Ceci permet de rendre plus complexe l'attaque car il ne suffit plus de deviner le mot de passe mais aussi le nom d'utilisateur. De plus, en interdisant la connexion en tant qu'administrateur, on impose l'utilisation de la commande SUDO pour les actions qui nécessitent des privilèges administrateur. Ceci a l'avantage d'offrir des *logs* complets des commandes exécutées et offre une meilleure traçabilité de ce qui se passe sur le serveur.

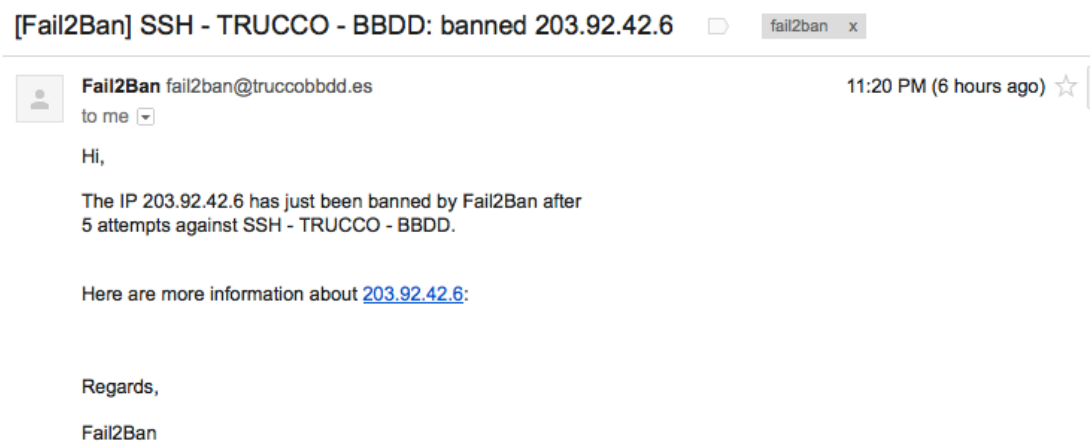
Finalement en utilisant un programme tel FAIL2BAN, on peut surveiller les tentatives de connexion qui échouent[20]. FAIL2BAN permet de surveiller n'importe quel type de log et de spécifier à l'aide d'expressions régulières ce qui représente une entrée suspecte ainsi que définir l'action à entreprendre. Cette flexibilité de configuration nous permet de surveiller non seulement le protocole SSH mais aussi les tentatives d'accès au portail d'administration de Magento.

Dès que l'on repère 5 tentatives de connexion échouée provenant de la même adresse IP, fail2ban rajoute une règle provisoire au pare-feu interdisant tout trafic provenant de l'adresse en question durant 15 minutes et notifie l'administrateur par courriel. Ceci combiné à des mots de passe de qualité ralentit les attaques au point qu' elles deviennent inefficaces[21]. Cette mesure n'est que modérément efficace contre des attaques distribuées mais dans le cas d'une réelle attaque ciblée de grande envergure, la notification par courriel permettra à l'administrateur de prendre les mesures nécessaires.

---

[20] Fail2Ban – http://fail2ban.org

[21] Microsoft: *Password must meet security requirements* - http://technet.microsoft.com/en-us/library/cc786468.aspx


[Fail2Ban] SSH - TRUCCO - BBDD: banned 203.92.42.6

fail2ban x

Fail2Ban fail2ban@truccobbdd.es 11:20 PM (6 hours ago)

to me

Hi,

The IP 203.92.42.6 has just been banned by Fail2Ban after 5 attempts against SSH - TRUCCO - BBDD.

Here are more information about 203.92.42.6:

Regards,

Fail2Ban

Détection d'intrusions :

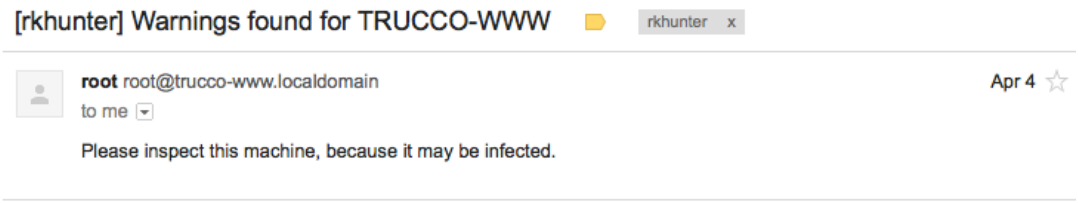
Il reste à envisager la possibilité que malgré nos efforts, le serveur soit compromis et il est impératif que nous le détections le plus vite possible pour en limiter l'impact.
RKHUNTER est un programme particulièrement efficace pour cette tâche, car il offre deux fonctionnalités particulièrement intéressantes[22].

- Analyse du système à la recherche de *ROOTKITS*
- Détection des changements de configuration

La recherche de *ROOTKITS* permet de détecter l'installation d'applications telles des chevaux de Troie qu'un attaquant pourrait utiliser pour contrôler le réseau de machine qu'il a réussi à infecter.

La détection des changements de configuration nous intéresse tout particulièrement, car une fois qu'un attaquant réussit à s'introduire dans le système, il est probable qu'il commencera par désactiver certaines des mesures de sécurité. Ces changements sont détectés par *RKHUNTER* durant son analyse quotidienne et notifie l'administrateur par courriel que le serveur a probablement été compromis.

[22] Rootkit Hunter Project - http://rkhunter.sourceforge.net



La sécurité d'un serveur est un tache complexe a mettre en œuvre, il est difficile voir impossible de connaître tous les vecteurs d'attaque et failles existant. Il est donc nécessaire un foi que la police de sécurité a été mis en place de la tester a l'aide d'outils spécialisés. NESSUS est un des outils les plus utilisé dans le monde de la sécurité, il permet d'effectuer une analyse de failles de sécurité et d'erreurs de configuration exhaustive. Son utilisation a rendu possible, entre autre, de détecter certaines erreurs de configuration subtile dans le protocole d'encryptions utilisé en HTTPS et de les corriger. Le rapport complet de Nessus est fourni en annexe et on peut y voir les avertissements qui mon permis de parachever l'installation.

Conclusion

Ceci conclut la première partie de mon stage, durant les 6 semaines dédier aux serveurs j'ai eu l'occasion d'apprendre a connaître un grand nombre d'outil et de services diffèrent. Ma formation a l 'IPL m'avait bien préparé à cette analyse particulièrement le cours « d'administration système » et j'ai eu l'opportunité de parfaire mes connaissances avec une expérience pratique.

Ayant construit l'architecture des serveurs j'ai aussi tout au long de mon stage du les administrer, ceci m'a permit de découvrir une partie des responsabilités d'un administrateur système ainsi que de continuer a étendre les fonctionnalités du serveur. Au fur et a mesure que les besoins du développement on évolué certains services tel que le contrôle de version on été rajouté pour faciliter le développement en collaboration avec les autres acteurs externe du projet.

La documentation des serveurs fut un point important et évolutif du stage dans le but de faciliter le transfert de mes responsabilités ainsi qu'aider a la configuration future d'autre projet similaire de la compagnie.

Tous les documents produits à cet effet ainsi que la configuration complète des serveurs sont disponible en annexe sur le support fourni avec ce travail.

Intégration ERP à MAGENTO

Élaboration d'ETL pour la communication avec SAP

Introduction :

Suite à la configuration des serveurs, la seconde partie du projet a été consacrée à l'intégration entre *SAP* et MAGENTO. Si l'E-Commerce est un nouveau marché pour *Trucco,* il est important que celui-ci s'intègre avec le système de gestion existant. MAGENTO offre une plateforme qui intègre des éléments de CRM et d'ERP mais ceci n'est pas une solution viable pour le client qui dispose déjà d'une infrastructure complexe et parfaitement adaptée à son modèle de vente. En outre, cela n'a pas de sens pour le client d'avoir à dupliquer le travail, car cela occasionne une augmentation des frais et de la complexité de gestion. Il convient donc d'arriver à importer les données préexistantes de manière à ce que l'e-Commerce s'intègre parfaitement à ses autres activités.

MAGENTO possède des outils d'importation qui permettent d'automatiser la création et la mises à jour des données (produits, clients…) mais dès les premiers tests, il est apparu que ces outils étaient inadéquats pour l'importation régulière de gros volumes d'information. Le souhait du client était que chaque nuit les données de SAP et MAGENTO soient synchronisées de manière à garder une uniformité des articles entre ses différentes plateformes de vente.

Actuellement *Trucco* dispose d'une dizaine de milliers d'articles. Un premier test d'importation avec les outils existants indique un temps d'exécution proche des neuf heures. Suite à l'optimisation des serveurs, ce temps s'est vu réduit à trois heures mais cela reste inacceptable, car durant cette période, le site perd fort en réactivité suite au processus intensif d'importation.

Il a donc été décidé d'implémenter des outils d'importation rapides et adaptés aux besoins spécifiques du client. Ceci est l'objet de la seconde partie de mon stage.

Les outils que je décris ici sont la description même de middleware, ils servent à acheminer l'information d'un système vers un autre, de manière à permettre la communication entre deux systèmes disparates. L'analyse des besoins a demandé énormément de rigueur, tant au niveau des spécifications de communication qu'au niveau de la robustesse requise pour garantir le bon fonctionnement de processus qui ne disposent pas de supervision humaine.

Magento en bref

En janvier 2007, la Société VARIEN, Agence Web spécialisée dans l'e-Commerce, débute le développement de MAGENTO. Depuis 2012, Magento est devenu la plate-forme d'e-Commerce la plus répandue[23].

MAGENTO s'appuie sur le Framework ZEND, qui apporte une architecture fonctionnelle standardisée et des fonctionnalités PHP étendues. MAGENTO est distribué sous licence open-source, rendant le code accessible aux développeurs du monde entier.

MAGENTO offre des fonctionnalités que l'on ne retrouve nulle part ailleurs, comme la possibilité d'administrer plusieurs sites e-Commerce depuis un seul et même *backend*. En outre, le logiciel offre une facilité d'administration exceptionnelle.

QUELLE EST LA PARTICULARITE DE MAGENTO?

Pour les magasins en ligne, Magento optimise le positionnement dans les moteurs de recherche. De plus son aspect open source et la création d'un marché de distribution d'extensions on amené a une grande disponibilité de contenu[24]. Ces extensions permettent entre autre l'intégration avec toute une gamme de services tel *Google Analytics*, *DHL*, *Twitter* etc..

Il existe de nombreux compétiteurs dans le domaine de l'e-Commerce sous différentes formes mais MAGENTO possède un nombre impressionnant de fonctionnalités offrant tout ce qui est nécessaire à la gestion d'un site de vente moderne. Un des atouts majeurs de la plate-forme est sa flexibilité: elle offre une interface CMS des plus complètes, permettant la génération quasi automatique de l'entièreté du contenu. Cette flexibilité est poussée à l'extrême par l'utilisation du modèle *Entity Atrribute Value* pour stocker les articles et les clients. Ce modèle permet, vu le CMS, de customiser à souhait les attributs des entités de MAGENTO. Un produit contient un nombre minimal d'attributs obligatoires tels que son nom et son prix, suivi d'un nombre illimité d'attributs créés par l'utilisateur pour refléter les besoins spécifiques de son business.

---

[23] SBWire : *Magento is the 2012's leader in e-Commerce* - http://www.sbwire.com/press-releases/sbwire-135966.htm

[24] Magento Connect - http://magentocommerce.com/magento-connect/

Situation de départ

A mon arrivée, le cahier des charges avait déjà été établi. Durant la première phase de mon stage consacrée aux serveurs, le client a commencé à élaborer l'exportation automatisée de ses données afin que je puisse commencer le travail d'importation. En parallèle mon équipe a entrepris les modifications nécessaires de Magento pour l'adapter au type de produit vendu par *Trucco* tant au niveau du backend qu'au niveau de la présentation.

Quand la seconde phase du projet a débuté, la plupart des ajustements nécessaires au développement des ETL avaient été effectués, de sorte que j'ai immédiatement pu me mettre au travail.

Les principales étapes ont été les suivantes :

- Mise en place du serveur FTP chez le client d'ou récupérer les données
- Développement SAP pour l'exportation journalière des données
- Personnalisation de Magento en fonction des articles de Trucco

Un des problèmes rencontrés au début du projet était le manque d'informations disponibles sur le fonctionnement interne de MAGENTO. Si la documentation de l'API proposé par le framework est extrêmement fournie et de bonne qualité, il n'en va pas de même pour ce qui est de son fonctionnement interne. A ce jour, je n'ai toujours pas trouvé une seule documentation de qualité expliquant en détail la structure des données utilisées par le framework, si ce n'est quelque diagramme rudimentaire de la base données. A cela s'ajoute que Magento a connu un nombre important de mises à jour, chacune apportant des changements de structure, ce qui a rendu l'analyse préalable difficile à établir.

Ainsi, au fur et à mesure de ma compréhension du fonctionnement interne de MAGENTO, j'ai dû a plusieurs reprises, apporter des changements à la spécification du format d'échange des données fournies par le client. Chacun de ces changements a dû être pris en compte par l'équipe de développement *SAP* pour finalement en arriver à la quatrième version finale de la spécification.

Analyse

Description générale

Le processus décrit par le cahier de charge pour l'importation des données est le suivant :

- Chaque jour, à heure fixe, une exportation de *SAP* est effectuée et placée sur le serveur FTP
- L'exportation se fait à l'aide du fichier XML et est séparée en 6 fichiers :

1. Articles configurables
2. Articles simples
3. Prix des articles configurables
4. Prix des articles simples
5. Stock des articles simples
6. Activation des produits

- Les fichiers importés doivent être archivés dans un dossier sur le serveur FTP
- Suite à l'importation des données un log pour chaque fichier importé est placé sur le serveur FTP décrivant l'exécution de l'importation
- A la fin de l'importation un log général de l'importation est placé sur le serveur FTP indiquant si un problème a été rencontré durant l'importation d'un des fichiers

Ceci décrit la structure de la charge journalière automatisée. A cela s'ajoute la possibilité pour le client de lancer l'importation d'un ou plusieurs de ces fichiers de manière indépendante depuis l'interface d'administration de MAGENTO.

Ceci a séparé le développement en deux phases, premièrement un utilitaire qui gère les transactions avec le serveur FTP et deuxièmement les outils d'importation pour chaque type de fichier.

Nous allons étudier deux types de produits offerts par MAGENTO:

- Les produits configurables, représentant des articles génériques par exemple une chemise à col croisé.

- Les produits simples, qui sont un affinement d'un produit configurable ayant deux attributs, la couleur et la taille. Exemple : chemise à col croisé vert de taille 36

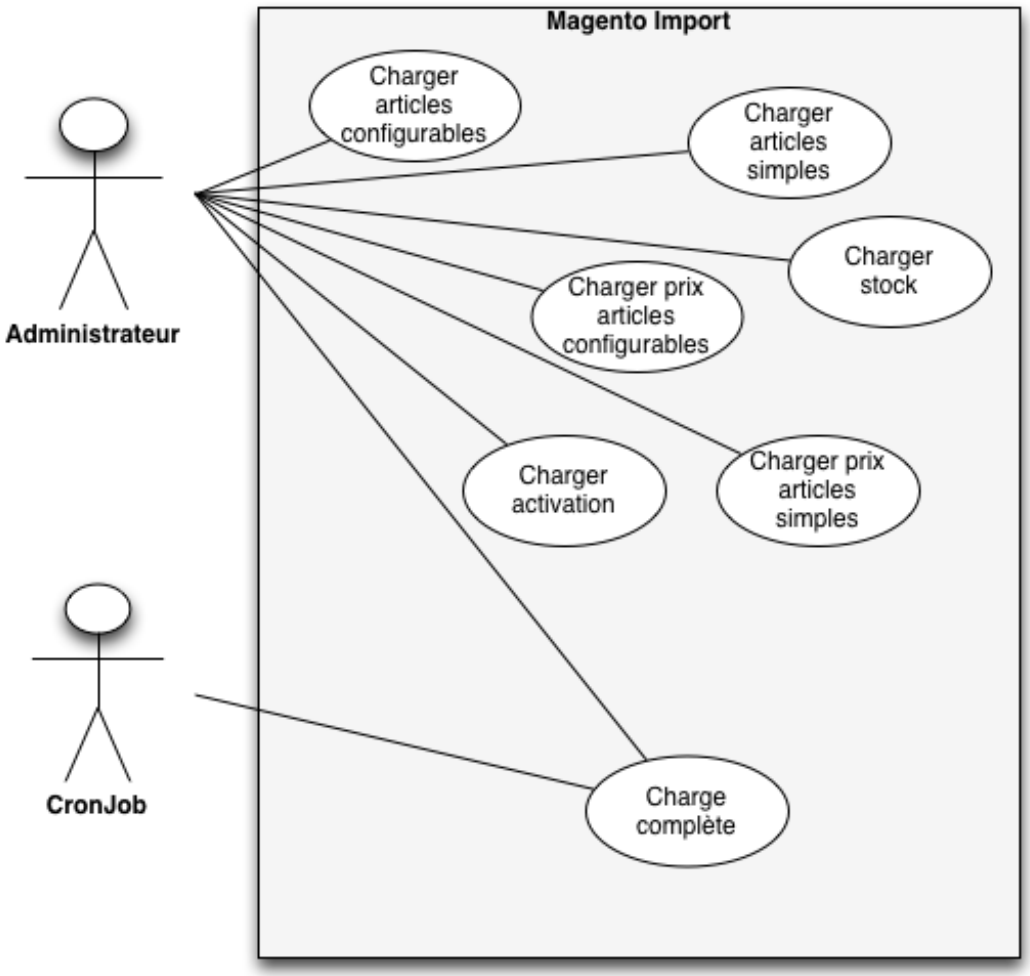
Le concept d'articles configurables est un des types génériques offerts par MAGENTO. Il permet de regrouper un nombre de produits simples en un seul article au client qu'il peut ensuite configurer selon les disponibilités.

Tous les articles que nous allons traiter doivent entrer dans une de ces deux catégories, c'est une des contraintes posées dans l'analyse. De plus, il ne peut exister un produit simple sans que celui-ci ne soit lié à un produit configurable, car seuls les produits configurables sont visibles pour l'utilisateur.

Cas d'utilisation

Il existe deux acteurs possibles :

- L'administrateur MAGENTO qui lance une ou plusieurs importations manuellement
- Le processus journalier (CronJob) qui se charge de l'importation totale



La possibilité pour l'administrateur du site de lancer l'importation de certains fichiers indépendamment est une fonctionnalité importante du site. En effet, si lors de la charge journalière une erreur est détectée, cela lui permet de relancer une importation sans avoir à attendre le jour suivant.

Cela peut être considéré comme un affinement du cas général d'importation journalière mais nécessite une interface graphique d'administration intégrée au site.

Choix du langage de développement

Le choix du langage de développement a été un sujet de débat au début du projet. Les problèmes de performance rencontrés avec les outils d'importation de Magento sont en partie liés au langage d'implémentation (PHP) mais aussi aux multiples couches d'abstraction. Il a été décidé de ne pas utiliser PHP pour les outils d'importation. PHP est un langage dont l'objectif est la présentation de contenu web et se prête mal aux besoins d'outils à haute fiabilité tel qu'un ETL.

Le choix le plus évident suite à ma formation aurait été Java, mais la mémoire vive du serveur web est précieuse et la machine virtuelle de Java a tendance à être gourmande. Un langage compilé tel C++ aurait été le plus avantageux au niveau des performances, mais il dépend fortement de la plateforme pour laquelle on développe et peut s'avérer délicat à gérer.

Le choix s'est posé sur Python : Python est un langage interprété mais qui offre des performances impressionnantes. Il offre une librairie standard extrêmement fournie et optimisée et est préinstallé sur la plupart des distributions de Linux. Si le fait de se lancer dans un nouveau langage de programmation est une tâche ardue, Python offre d'énormes avantages en matière de vitesse de développement et de lisibilité du code. Python fut une réelle découverte au cours de mon stage et est rapidement devenu mon langage de prédilection.

Méthodologie de travail

J'ai abordé brièvement les problèmes rencontrés avec la structure des données de MAGENTO et ce fut un problème récurrent tout au long du projet. La seule manière de comprendre réellement ce qui se passait à l'intérieur de MAGENTO a été de travailler avec des dumps de bases de données. Ce procédé est extrêmement laborieux mais a le mérite d'être complet. En utilisant un outil de comparaison de fichier tel *kdiff3* pour comparer l'état de la base de données avant et après la création d'un produit, il a été possible de voir chaque modification apportée au système[25].

MAGENTO se base sur un modèle appelé Entity Atrribute Value [26]. Ce modèle a l'avantage de découpler entièrement un article des éléments qui le compose. Ceci permet une grande flexibilité et permet d'adapter MAGENTO à tout type de business. Mais ce qui fait la force de ce modèle est aussi ce qui en fait sa faiblesse. Il n'existe pas de table représentant un article dans son entièreté, chaque attribut d'un article est stocké comme une entrée dans une table générique. Pour donner une idée de ce que cela implique, il faut savoir qu'un article standard a en moyenne une quarantaine d'attributs. Récupérer un article consiste à faire autant de requêtes à la base de données qu'il y a d'attributs. Pour résoudre ce problème MAGENTO a implémenté des tables appelées *flats file* où les articles sont stockés comme une seule grande entrée texte. Si cela permet de pallier les problèmes de performance, cela crée énormément de duplication d'information. Au final l'insertion d'un seul produit implique près d'une trentaine de tables qui ne sont pas homogènes et utilise des conventions de codages différentes.

Pour donner une idée de l'ampleur du problème au lecteur voici un diagramme de base de données de MAGENTO qui compte plus de 220 tables :

[25] Kdiif3 - *http://kdiff3.sourceforge.net*
[26] Wikipedia : *Entity attribute value model* - http://en.wikipedia.org/wiki/Entity–attribute–value_model

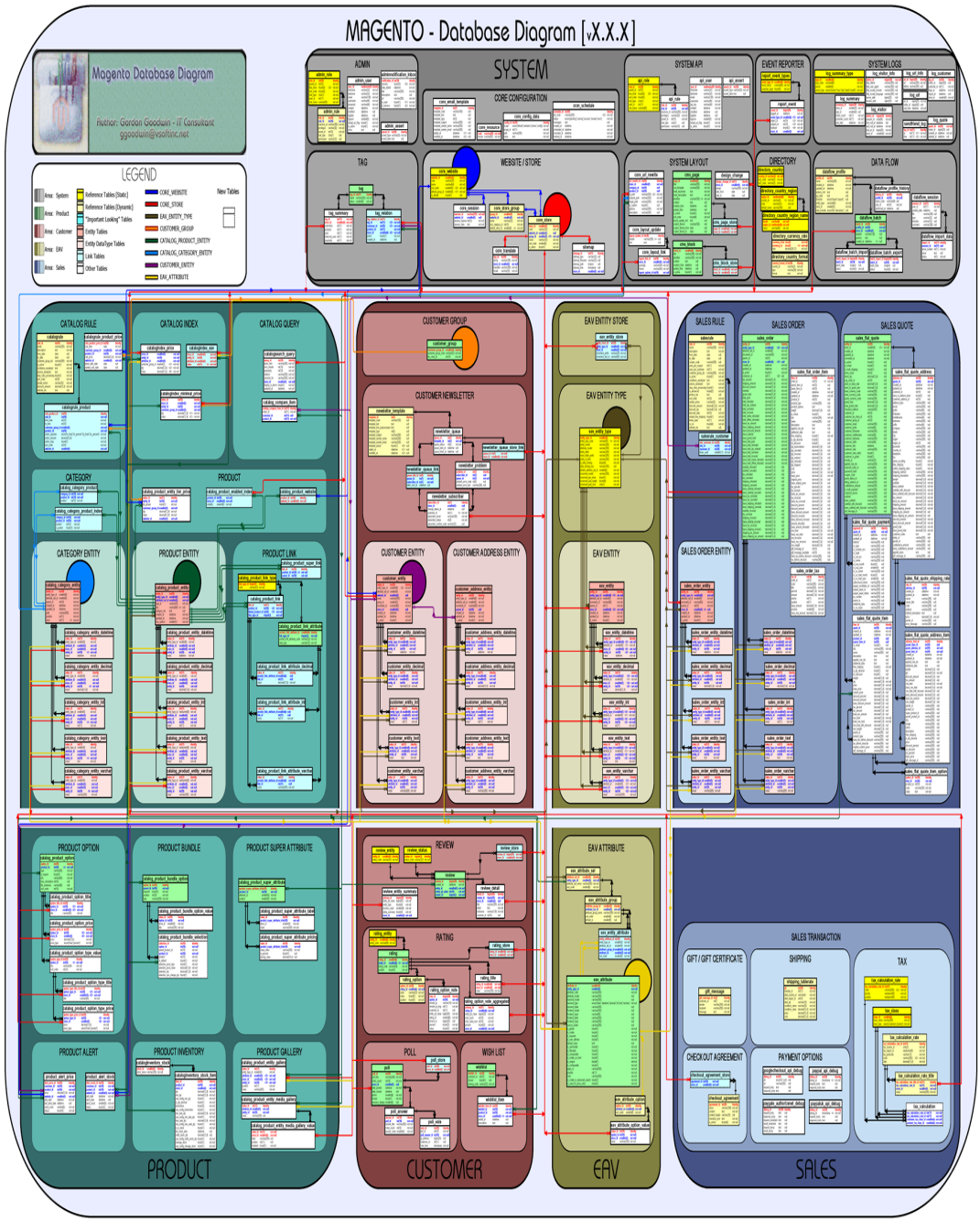


Diagramme de base de données MAGENTO source :
http://www.MAGENTOcommerce.com/wiki/2_-_MAGENTO_concepts_and_architecture/MAGENTO_database_diagram

Élaboration des outils d'importation

Une fois l'analyse de MAGENTO et la spécification des formats d'échange de données établie, le développement des ETL a pu commencer. Il y a trois phases à chacun des outils d'importation :

1. Extraction des données des fichiers XML
2. Transformation des données vers le format de Magento
3. Chargement dans la base de données

Un effort a été fait pour découpler l'implémentation des ETL de la représentation des articles. Etant donné que Magento offre une grande souplesse au niveau de la configuration des articles, il est possible que durant le cycle de vie de la plateforme le client désire ajouter ou modifier les attributs de ses articles.

MAGENTO fait la distinction entre deux types d'attributs d'un produit, ceux qui respectent le modèle EAV et qui peuvent à tout moment être ajoutés ou retirés à une catégorie d'articles (*p. ex.*: la taille d'un vêtement ou la température de repassage) et les attributs obligatoires qui sont communs à tous les articles (*p. ex.*: le nom, la description, le prix…). Cette structure est extrêmement flexible et permet l'évolution du site. Un effort a donc été fait pour permettre que les ETL soient paramétrables à l'aide de fichiers de configuration et puissent prendre en compte ce type de changement sans modification du code.

Pour ce faire, un langage intermédiaire a été développé qui décrit les fichiers XML ainsi que leur relation avec la base de données. Ceci a rendu le procédé d'importation plus complexe car on ne connaît que peu de la structure des fichiers qui nous sont envoyés. A cela s'ajoute le besoin de vérifier la validité des données que nous importons. Comme nous n'utilisons pas l'API de MAGENTO et traitons directement avec la base de données, nous perdons toute vérification d'erreurs et cela peut se révéler catastrophique.

Ces fichiers de configuration fournissent la majorité de l'information nécessaire à l'importation, ils indiquent pour chaque attribut d'un article :

- le nom du tag XML
- le nom en base de données
- si l'attribut est obligatoire
- le type d'EAV

- une valeur par défaut si le champ n'est pas renseigné

Voici un exemple simplifié de fichier de configuration pour un produit configurable.

```
# DBNAME                    : XML_TAG_NAME      : RECURSIVE[FILE]   : REQUIRED[0-1] : EAV[0-3]   : DEFAULT
sku                         : MATERIAL          :                   : 1            : 0          :
entity_id                   :                   :                   : 0            : 0          :
has_options                 :                   :                   : 1            : 0          : 1
required_options            :                   :                   : 1            : 0          : 1
entity_type_id              :                   :                   : 1            : 0          : 4
attribute_set_id            :                   :                   : 1            : 0          : 11
type_id                     :                   :                   : 1            : 0          :
configurable_descripti      : DESCRIPCION       : description       : 1            : 1          :
name                        : NOMBRE_PRODUCTO   : name              : 1            : 1          :
trucco_medida               : MEDIDA            : measurement       : 0            : 1          :
trucco_destacado            : DESTACADO         : announce          : 0            : 1          :
keywords                    : KEYWORDS          : keywords          : 0            : 1          :
cross_sales                 : VENTA_CRUZADA     : cross_sale        : 0            : 0          :
short_description           : DESCRIPCION_CORTA : short_description : 1            : 1          :
trucco_composicion          : DESC_COMPOSICION  : composition       : 0            : 3          :
generic_parent              : GENERICO          :                   : 0            : 0          :
article_group               : GR_ARTICULOS      :                   : 0            : 0          :
trucco_color                : COLOR             :                   : 0            : 4          :
trucco_talla                : TALLA             :                   : 0            : 4          :
trucco_ano                  : ANO               :                   : 0            : 1          :
trucco_temporada            : TEMPORADA         :                   : 0            : 2          :
trucco_lavado               : LAVADO            :                   : 0            : 2          :
trucco_secadora             : SECADORA          :                   : 0            : 2          :
trucco_plancha              : PLANCHA           :                   : 0            : 2          :
trucco_lejia                : LEJIA             :                   : 0            : 2          :
trucco_tinte                : TINTE             :                   : 0            : 2          :
trucco_tendido              : TENDIDO           :                   : 0            : 2          :
status                      : ACTIVO            :                   : 0            : 1          :
price                       :                   :                   : 1            : 1          : 0
weight                      :                   :                   : 1            : 1          : 0
visibility                  :                   :                   : 1            : 1          : 4
tax_class_id                :                   :                   : 1            : 1          : 0
options_container           :                   :                   : 0            : 1          :
```

Sur base de ces fichiers de configuration, il a été possible de construire des objets paramétrables en utilisant l'introspection. Ces objets ont des lors été emballés dans un objet de type proxy qui permet de les consulter selon différentes interfaces en fonction du traitement qu'on est en train d'effectuer[27].

On a deux interfaces :

- Une pour la partie extraction du XML qui permet d'indiquer au parseur comment interpréter le fichier et stocker le contenu

- Une autre pour l'importation vers la base de données qui renseignent les noms des champs et leur type

Ceci a permis de garder les deux contextes de traitement séparés tout en gardant un objet commun qui contient les données. A ceci est venu s'ajouter une implémentation du pattern flyweight qui optimise la création des objets sans avoir à parcourir les fichiers de configuration[28].

Il fut important de produire une documentation du fonctionnement de ces fichiers de configuration, car ils contiennent des éléments de logique métier et seront amenés à être réutilisés pour d'autres projets.

Chaque ETL a été développé de manière à pouvoir être exécuté indépendamment. Ce sont des programmes en ligne de commande. Ils respectent tous la même structure d'appel de manière à ce que le procédé d'exécution puisse être automatisé. Cette structure est la suivante : programme.py fichierDeConfigurationMagento fichierXMLaParsé nomDuLog

Le fichier de configuration de MAGENTO renseigne la base de données et les informations de connexion nécessaires. Voici un exemple :

```
IP : trucco.database
User : truccotest
Password : XXXXXXX
Schema : truccotest
```

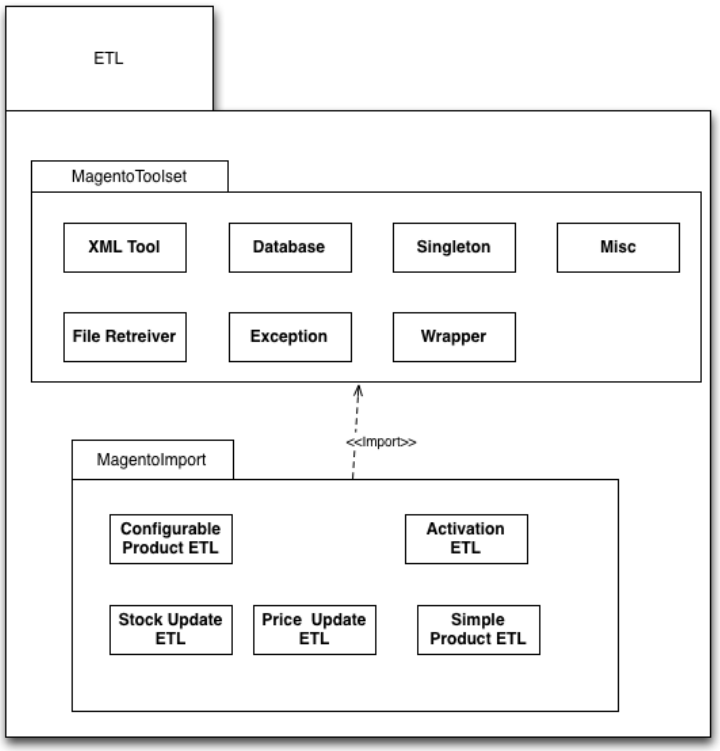
Un des problèmes rencontrés au long du développement est la qualité des logs. Comme le procédé est entièrement automatisé, il est important d'indiquer très précisément pourquoi certains articles sont incorrects en ne peuvent être importés. En jouant avec l'introspection

[27] Wikipedia: *Proxy Pattern* - http://en.wikipedia.org/wiki/Proxy_pattern
[28] Wikipedia: *Flyweight Pattern* - http://en.wikipedia.org/wiki/Flyweight_pattern

il a été possible d'indiquer ce qui est incorrect avec un certain niveau de fiabilité mais cela reste délicat et c'est un des éléments du programme qui reste à améliorer.

Le code a été séparé en deux paquets, l'un contenant les différents ETL pour les 6 types de fichier et le second contenant les utilitaires tels le parseur XML ou les objets d'accès à la base de données qui sont partagés par chaque ETL. Ceci permet de limiter la duplication de code et de mieux le structurer.

Voici le diagramme de paquet :



A la fin de développement une semaine a été consacrée aux tests d'intégration. MAGENTO agissant comme une boite noire il n'était pas possible d'automatiser ces tests et ils ont été réalisés conjointement avec le client. Durant cette semaine certains correctif on été amené au projet suite aux remarques émise par le client. Cette coopération a permis de mieux comprendre certains détails des articles de Trucco et de s'assurer que l'implémentation était conforme aux besoins.

Utilitaire de transfert de fichiers

Suite au développement des ETL la seconde tache a été de développer un outil qui automatise charge le transfert des fichiers entre les serveurs FTP ainsi que de l'intégrer avec MAGENTO.

Dans le cahier des charges, il a été établi que chaque jour les fichiers à importer sont placés dans un répertoire sur un serveur FTP du client. Pour séparer les outils d'importation du protocole de transfert un outil a été développé qui se charge du :

1. Téléchargement des fichiers
2. La vérification de leurs intégrités
3. L'exécution de l'outil d'importation leur correspondant
4. L'exportation des logs sur le serveur FTP
5. L'archivage des fichiers importés sur le serveur FTP

Une convention a été établie avec le client sur la nomenclature des fichiers XML ainsi que des logs. L'exportation est journalière, nous avons donc opté pour ce format : NomDuFichierAAAAMMJJ.xml ou AAAAMMJJ représente l'année, mois et jour de l'importation

Rapidement on a ajouté la vérification d'intégrité des fichiers par signature MD5, parce que les fichiers sont transférés sur internet et que le protocole TCP ne garantit pas leur intégrité. Ceci est un compromis par rapport à la proposition faite au client : initialement, il avait été proposé en plus de la signature MD5 d'encrypter les fichiers avec une encryption à clef publique[29]. Ceci aurait permis d'assurer que les fichiers ont bien été produits par *Trucco*, même en cas de faille de sécurité au niveau du serveur FTP mais cela a été jugé trop complexe par le client.

L'outil a été développé de manière à pouvoir être utilisé dans d'autres contextes du projet qui nécessitent une interaction avec un serveur FTP. Pour ce faire on utilise des fichiers de configuration qui définissent les fichiers à récupérer et le programme qui doit être appelé :

```
#            FILE                SCRIPT                          LOG
MAESTROS_YYYYMMDD.xml  :  importDatosMaestros.py     :  MAESTROS_YYYYMMDD.log
GENERICO_YYYYMMDD.xml  :  configurableProductImp.py  :  GENERICO_YYYYMMDD.log
```

[29] Wikipedia : *Public-key Cryptography* - http://en.wikipedia.org/wiki/Public-key_cryptography

Cet outil a aussi pour but d'indiquer une éventuelle erreur dans le processus importation et le cas échéant d'indiquer dans quel log se trouve la description de l'erreur :

```
2012-05-25 11:26:08,654 - INFO:Script importDatosMaestros.py executed
correctly.
2012-05-25 11:26:19,257 - WARNING:Error in downloading the remote file :
DATOS_MATERIALES.xml. File not found.
```

Pour intégrer cet outil avec MAGENTO nous avons fait usage des tâches planifiées ou MAGENTO Cron jobs. Comme l'outil Cron de Linux, il permet d'enregistrer des tâches à exécution régulière. De plus, nous allons utiliser l'interface d'administration de Magento pour permettre l'exécution des importations par l'utilisateur à la demande.

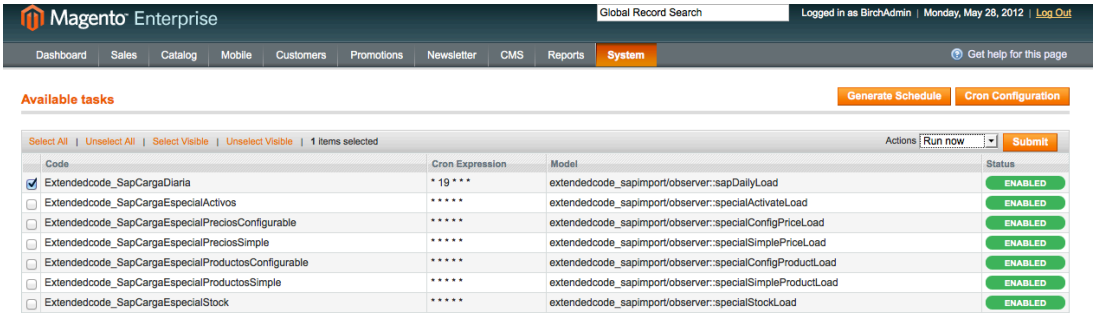
Pour ce faire, il faut étendre l'observer de MAGENTO ainsi que configurer l'exécution planifiée :

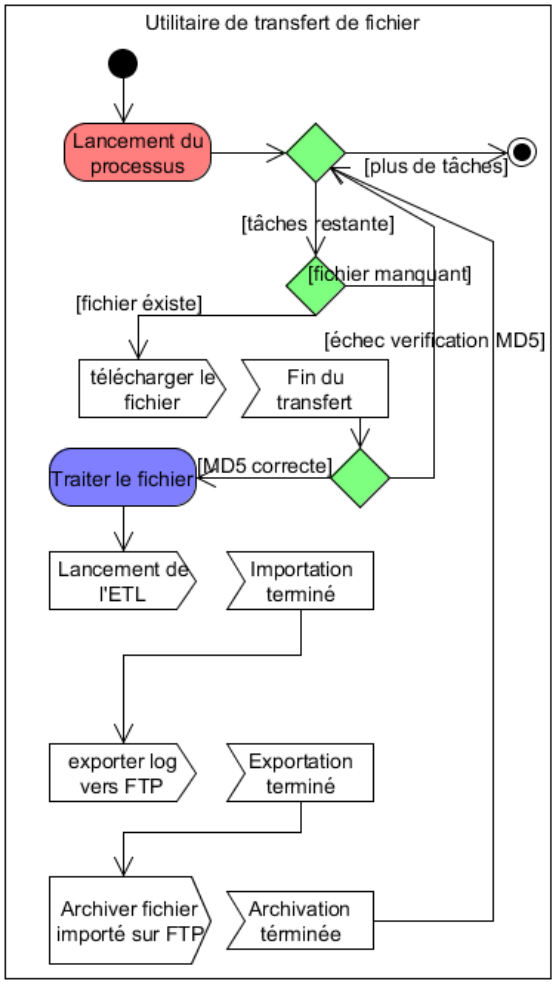
```
<?xml version="1.0"?>
 <config>
        <modules>
          <Extendedcode_Truccoloads>
            <version>0.0.1</version>
          </Extendedcode_Truccoloads>
        </modules>
    <global>
        <models>
          <extendedcode_truccoloads>
            <class>Extendedcode_Truccoloads_Model</class>
          </extendedcode_truccoloads>
        </models>
    </global>

    <crontab>
        <jobs>
          <Trucco_loads_sap_CargaDiaria>
          <schedule><cron_expr>0 2 * * *</cron_expr></schedule>

          <run>
           <model>
             extendedcode_truccoloads/observer::sapDailyLoad
           </model>
         </run>
        </Trucco_loads_sap_CargaDiaria>
       </jobs>
   </crontab>
 </config>
```

Nous pouvons voir ici que la charge journalière est exécutée à deux heures du matin avec l'expression 0 2 * * * et que la fonction à exécuter est sapDailyLoad se trouvant dans le paquet extendedcode dans la classe truccoloads qui étend un observer. Ceci est un des

fichiers de configuration typiques de MAGENTO. L'observer permet de faire le lien entre MAGENTO et les ETL et permet d'indiquer à l'utilisateur le résultat des importations.

L'interface de contrôle se présenté comme suit :





Le flux d'exécution de l'utilitaire d'importation ce présente comme ceci.

Cet utilitaire et soit lancé par un utilisateur au travers de l'interface d'administration de MAGENTO soit exécuté automatiquement par le planificateur de tâche.

Dans le premier cas, la valeur de retour de l'utilitaire est utilisée pour communiquer le résultat de l'exécution à l'utilisateur.

Dans le second cas, seul les logs exporté sur le serveur FTP indique le résultat de l'exécution.

Conclusion

Durant le stage j'ai eu l'occasion d'approfondir certain aspect théorique des enseignements de l'IPL en étant confronté à un développement dans un milieu hautement professionnelle. Ces trois mois mon beaucoup appris tant au niveau des techniques et outils de programmation mais aussi au niveau relationnel. La communication entre les membres de l'équipe et la bonne documentation c'est relevé un point essentiel du stage.

Mon chef de projet fut d'une aide précieuse, les meeting d'équipe on permis de synchroniser le développement et était l'occasion d'exprimer les difficultés rencontrée et de demander conseil. Si la communication en espagnole fût parfois délicate, j'ai reçu le soutient et les conseils de mes collaborateurs ce qui m'a permit de mener à bien ce stage.

La première partie du projet fut un long travail de recherche, il était important de savoir démontrer la validité de ma solution et de la documenter de manière a ce qu'elle soit réutilisable dans des futurs projets. Elaborer la sécurité fut une responsabilité impressionnante du au conséquences possible d'une erreur mais très éducative. Cela m'à permis le fonctionnement des services de Linux mais m'a offert une bien meilleure compréhension du système d'exploitation.

L'objectif d'élaborer des serveurs pour un environnement de production est pour moi réussi, apprendre à évaluer mes configurations en utilisant des outils de mesure adaptée fut une étape importante et m'a permis d'établir un niveau de confiance dans ma solution. Suite à cela j'ai eu l'occasion de participer brièvement à un audit pour un autre projet de la compagnie qui présentait des aspects similaire au mien. Ce fut une expérience intéressante qui m'a permit de mettre en pratique mes nouvelles connaissances dans un autre contexte.

La seconde partie du projet m'a surpris par sa complexité, le temps nécessaire à l'analyse fut beaucoup plus long que prévu et a offert des épreuves conséquentes. Les exigences du projet m'ont poussés à aborder Magento à l'envers. Je me suis familiarisé avec le framework pour mieux comprendre son fonctionnement et développer les interfaces nécessaires au projet mais la majeure partie du développement a été liée à la structure interne de la plateforme. Le manque de documentation et la structure complexe on demandé beaucoup de patience et de méticulosité mais le travail a été récompensé par la mise en production et l'approbation du client.

Ce stage m'a permis de découvrir la vie d'entreprise et j'envisage ma carrière professionnelle avec enthousiasme.

Annexes

Les annexes son fournies sur un support USB qui accompagnent ce travail.

Sur ce support vous trouverez trois dossiers intitulés:

1. Mémoire – Contient le travail de fin d'études sous plusieurs formats électronique.
2. Partie 1 – Contient tout ce qui est en relation avec la partie serveurs de ce travail.
3. Partie 2 – Contient tout ce qui est en relation avec le développement des ETL.

Le répertoire sur les serveurs contient :

1. La documentation des configurations produite pour la compagnie (en anglais)
2. La configuration complète de chaque serveur
3. Les scripts écrit durant mon stage
4. Les tests de sécurité et d'optimisation

Le répertoire sur les ETL contient :

1. La documentation des ETL (en anglais)
2. La documentation de l'outil de transfert de fichier (en anglais)
3. Le code développé durant mon stage
4. Les tests unitaires
5. Tous les documents d'analyse du projet (en Espagnole) avec un document qui indique mes contributions.

Les deux dossiers relatifs au travail effectué durant le stage sont structurés en 8 sous dossiers qui sépare les différents éléments énoncé ci-dessus.

A la racine du support vous trouverez un document explicatif qui reprend en plus de détails les informations présentées ici et des informations complémentaire sur la part de contribution.

Agenda

Première partie du stage

- Semaine 1- Rencontre avec l'équipe, lecture du cahier de charges et découverte de Magento.
- Semaine 2- Réception des serveurs, installation de Magento et mise en place des l'environnement de travail.
- Semaine 3- Optimisation du serveurs web et installation des services de contrôles de versions.
- Semaine 4 – Suite des optimisations, installation de services supplémentaire sur les serveurs.
- Semaine 5- Fin de l'optimisation du serveur web et sécurisation.
- Semaine 6- Configuration, optimisation et sécurité du serveur de base de donnés.

Seconde partie du stage

- Semaine 7- Étude du fonctionnement interne de Magento, écriture de la première version de la spécification XML.
- Semaine 8- Implémentation du premier ETL pour l'importation de produit configurables et installation de plugins Magento pour le front end.
- Semaine 9- Révision de la spécification XML, Implémentation de l'ETL pour produit simple.
- Semaine 10- Implémentation des ETL pour l'importation des stocks, prix et activation.
- Semaine 11- Implémentation de l'outil et mise en production de l'outil de transfert FTP.
- Semaine 12- Tests d'intégration avec le client, implémentation de correctifs pour les erreurs détecté.
- Semaine 13- Documentation des serveurs et écriture de procédure standard pour les ETL.
- Semaine 14- Rédaction du mémoire

Glossaire

LAMP : LAMP est un acronyme désignant un ensemble de logiciels libres permettant de construire des serveurs de sites web. L'acronyme original se réfère aux logiciels suivants :

Linux, Apache, MySQL et PHP

Source : Wikipedia

Dépôt (repository) : Un dépôt au sens Linux, est une source centralisé contenant des logiciels d'ou ils peuvent être récupéré, installé et mis a jour de manière automatisée.

Virtual Host : Virtual host ou hébergement virtuel est une méthode permettant d''héberger de multiple nom de domaine sur une même machine.

Licence BSD : Licence BSD (*Berkeley software distribution license*) est une licence libre utilisée pour la distribution de logiciels. Elle permet de réutiliser tout ou une partie du logiciel sans restriction, qu'il soit intégré dans un logiciel libre ou propriétaire.

Source : Wikipedia

Cache : Technique utilisé en informatique qui permet d'éviter l'accès récurrent a une source de données en effectuant une copie des informations lors de la consultation initiale.

CSV : *Comma-separated values*, connu sous l'acronyme CSV, est un format informatique ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules.

Source : Wikipedia

Keep-alive : Méthode permettant de garder une connexion TCP ouverte entre de multiple requête émise par un client. Permet évite de devoir recréer une nouvelle connexion pour chaque requête émise.

White listing (Liste Blanche): Définition d'un ensemble d'éléments a qui on donne des autorisation dans un système, ce qui n'est pas précisé est considéré comme étant interdit.

Black listing (Liste noire): Opposé de liste blanche, définition d'éléments interdit dans un système, ce qui n'est pas stipuler est considéré comme autorisé

Opcode: Suite de bits représentant des opérations pour un processeur pour l'execution d'un programme informatique.

Brute force: Technique de hacking consistant a essayé toute ou un sous ensemble des combinaisons possible de valeur en cherchant à pénétrer un système informatique ou une méthode d'encryptions.

Botnet: Réseau de machine infecté et sous le contrôle d'une entité centrale qui dirige leurs actions.

Rootkits: Programme qui offre des fonctionnalités étendues de par l'utilisation ou la modification de fonctionnalités du noyau du système d'exploitation. Souvent utilisé pour décrire des malware qui compromettent un système et qui tente de maintenir l'accès au système en modifiant ces fonctionnalités.

Middleware: Programme ajoutant des fonctionnalités l'interconnexion de deux applications autrement indépendante.

CMS: Content Management System, terme générique décrivant des outils qui servent à aider l'édition et la publication de contenu sur une plateforme en regroupant de manière simplifié.

Dumps de base de donnée: Un dump d'une base de donnée correspond a l'exportation sous un format texte de l'ensemble du contenu d'une de données. Souvent utilisé pour la sauvegarde des informations qu'elle contient.

ETL: Extract, Transform & Load, terme générique qui décrit des outils dont l'objectif est l'acheminent de données d'un système a un autre et qui nécessite un étape de conversion des données pour les rendre compatible.

MD5: Algorithme de hachage cryptographique qui permet d'obtenir la l'empreinte numérique d'un fichier.

Source : Wikipedia

Bibliographie

BOTELHO Stefanie *Launching (and maintaining) a successful e-commerce platform: how publishers are creating and managing their digital retail operations*, - the Magazine for Magazine Management, 2011, Vol. XL (12), p. 22. - http://www.magentix.fr

DE LATTRE A., GARRIGUE R., TANGUY O., GRAND A. Formation Debian GNU/Linux. 2002-2011, - http://formation-debian.via.ecp.fr/

DOWNEY Allan B. Think Python: How to Think Like a Computer Scientist. [Cambridge University Press] Cambridge, 2009.

HUSKISSON J. Magento 1. 3: PHP Developer's Guide. [PACKT Publishing] Birmingham, 2010.

LUTZ M. Programming Python, 4th Edition - Powerful Object-Oriented Programming. [O'Reilly Media] Sebastopol CA, 2010.

McCOMBS A., BAHN, R. The Definitive Guide to Magento: A Comprehensive Look at Magento. - [Apress] Berkeley CA, 2010.

MØLLER Ch., CHAUDHRY, S. Re-conceptualizing Enterprise Information Systems (*5th IFIP WG 8.9 Working Conference, CONFENIS 2011, Aalborg, Denmark, October 16-18, 2011*) - Lecture Notes in Business Information Processing, 2012, Vol. 105, Part 2, pp. 64-74. - http://www.springerlink.com/content/p181022060250439/

RATSIATANDRA Haingosoaniony Magento: Réalisez des développements professionnels avec PHP - [Collection: Expert IT - Editions ENI] - Saint-Herblain, 2012. - http://books.google.nl/books?hl=nl&lr=&id=LtW59HV8WVcC&oi=fnd&pg=PP1&ots=Jponfu3tEK&sig=It9sL

ZENNER R., NEITZEL R. Online-Shops mit Magento-2. [O'Reilly Vlg. GmbH & Co.] Köln, 2011.