

### Document présentation

<b>Description</b>	Cahier des tests ProxyStat-Gestion
--------------------	------------------------------------

### Document certification

	<i>Name</i>	<i>Fonction</i>	<i>Date livraison</i>
<i>Author</i>	RAKOTOBÉ Eric	Responsable qualité	3 avril 2013
<i>Decidor</i>	CRASKE Antoine	Chef de projet	4 avril 2013

### Document Version history

<i>Version</i>	<i>Date</i>	<i>Change Summary</i>	<i>Change Author</i>
1.0	19 février 2013	Initialisation du document	RAKOTOBÉ Eric
2.0	3 avril 2013	Mise à jour au rendu V2	RAKOTOBÉ Eric
3.0	04 juin 2013	Mise à jour au rendu V3	RAKOTOBÉ Eric

# Objectif du document

## Table des matières

[Objectif du document](#)

[Table des matières](#)

[Méthodologie](#)

[Tests unitaires](#)

[Tests d'intégration](#)

[Tests de performances](#)

[Tests de sécurité](#)

[Tests exploratoires](#)

[Périmètre couvert](#)

[Revue de code](#)

[Améliorations](#)

[Configuration](#)

[Tests de performances](#)

[Scénario page d'accueil](#)

[Scénario authentification valide](#)

[Scénario authentification invalide](#)

[Enregistrement des résultats](#)

[V1 : 22 février 2013](#)

[Tests unitaires](#)

[Tests d'intégration](#)

[Tests de performances](#)

[V2 : 26 mars 2013](#)

[Tests unitaires](#)

[Tests d'intégration](#)

[Tests de performances](#)

[V3 : 7 juin 2013](#)

[Tests unitaires](#)

[Tests d'intégration](#)

[Tests de performances](#)

[Mesures avec l'outil Sonar](#)

[Tests multi navigateur](#)

[Tests de sécurité](#)

[Résultats des tests automatisés](#)

[Logs de l'outil Wapiti](#)



# Méthodologie

Il n'est pas possible de tout tester dans un projet.

Les paragraphes suivants présentent les choix que nous avons pris afin d'obtenir le meilleur rendement en terme de qualité logicielle.

Nous utilisons l'outil Sonar afin de suivre les différents indicateurs mentionnés.

Sonar est installé sur le serveur d'intégration continue mis à notre disposition, ce qui dispense les développeurs de lancer l'analyse en local.

## Tests unitaires

Les tests unitaires nous permettent de valider le fonctionnement d'un composant en isolation. Nous utiliserons des mécanismes de mocking afin de simuler l'environnement autour d'un composant pour assurer le testing en isolation.

Nous avons fait le choix d'avoir une classe gérant l'accès aux données de manière générique, cela nous permettant d'avoir toutes les opérations standards (CRUD, count) en étendant cette classe dans n'importe quel DAO. Nous avons donc testé entièrement ces méthodes car étant utilisées partout nous devons en assurer la robustesse.

Au niveau de la base de données nous utilisons un script qui rafraîchit les données à chaque passage des tests via maven. Cela nous permet d'assurer une qualité de jeux de données fiable et une reproduction viable des tests. Nous n'utilisons pas de base de données en mémoire pour tester sur la même base que celle de production Postgresql afin d'éviter toute surprise.

Le nommage des tests unitaires respectent les standards. Toute méthode de test est préfixée par 'test' et doit décrire en anglais l'action à réaliser et dans quelle condition. Par exemple "testBookingServiceThrowsExceptionWhenGuestsAreLessThanOne()". Cela nous évite d'avoir à gérer des commentaires expliquant le but du test, nous avons une documentation vivante des fonctionnalités en regardant tous les noms des tests exécutés. Ceci est bien sur valable à condition de conserver un taux de couverture suffisant.

Les tests unitaires sont rédigées en trois blocs de code : given, when, then. Le premier décrivant le contexte du test avec les variables et initialisation du contexte, le when qui effectue l'action proprement dite par exemple appeler un service, et le then vérifie l'état attendu à la fin du test, c'est donc l'étape de vérification.

Le taux de couverture défini est de 100% sur la couche DAO et de 80% sur la couche de service. Nous ne pouvons en effet pas tout tester - les 20% non testés sur la couche service sont les services les moins utilisés et les moins critiques pour la société ProxyStation qui seront les méthodes appelant uniquement les DAOs sans aucun traitement particulier. Nous effectuons également une revue des tests lors des spécifications pour se focaliser sur les cas de tests et pas uniquement la couverture du code. On peut par exemple tester quatre fois le même code mais avec des paramètres différents pour vérifier le comportement.

Les tests unitaires sont effectués avec le framework standard Java JUnit en version 4. L'environnement est mocké avec le framework Mockito. Ces deux outils sont des standards du marché qui ont fait leur preuve. Ils sont automatisés avec maven.

Les métriques suivies sont le taux de couverture de code par branche de code, car le taux de couverture par ligne n'est pas fiable étant donné qu'il prend en compte le nom des classes. Évidemment nous suivons le taux de réussite des tests qui doit être de 100%. Aucun changement dans le gestionnaire de versions n'est poussé par un développeur si le périmètre complet de tests n'est pas validé sur la machine. Ceci pour éviter les bugs en cascade et aussi qu'un développeur ne bloque ou ralentisse le développement des autres membres de l'équipe.

## Tests d'intégration

Les tests d'intégrations nous permettent de valider le fonctionnement des composants entre eux. Nous avons fait le choix d'utiliser un outil permettant de simuler des actions sur un navigateur web afin d'être au plus près de la vision utilisateur et assurer l'intégration de bout en bout. Les tests d'intégrations sont tous automatisés et doivent être validés pour packager l'application et la déployer sur les serveurs d'applications. Nous avons mis cette contrainte dans l'outil maven utilisé pour gérer le cycle de vie de l'application.

Nous utilisons l'outil Selenium (dernière version avec WebDriver) avec le driver Firefox. Les tests de compatibilité multi-navigateurs sont eux effectués avant chaque livraison. Nous avons fait ce choix pour garder un temps d'exécution de ces tests dans une limite de 5 minutes maximum en parallélisant. Nous ne pouvons donc pas exécuter les tests sur tous les navigateurs à chaque changement de code pour des raisons de délais de livraisons, et que les développeurs garde un feedback rapide sur leurs modifications.

Nous mesurons les tests d'intégrations par leurs taux de succès qui doit être à 100%. Nous validons également que les tests implémentés correspondent bien à ceux présents dans le document de spécifications détaillées et que nous passons bien par tous les écrans.

## Tests de performances

ProxyStation n'a pas été en mesure de nous fournir le trafic prévisionnel sur leur site internet. Nous nous sommes donc basé sur le nombre actuels de clients en appliquant le taux de transformation moyen des sites web de 2%.

ProxyStation gère environ 1500 réservations par an, ce qui nous donne un trafic prévisionnel annuel de 75 000 visites soit 205 visiteurs uniques par jour. Détails du calculs ci-dessous :

1500 réservations = 2% du trafic

75 000 visiteurs uniques annuels = 100% du trafic

Afin de garantir une solution robuste et évolutive, Dream IT a décidé d'assurer des temps de réponse stables inférieurs à 1 seconde pour un trafic annuel double de 150 000 visites.

Les tests de performances sont réalisés de façon incrémentale en enregistrant les résultats, pour être capable de détecter un seuil de non-performance. Ils sont également effectué pendant plus de 30 minutes pour assurer la fiabilité de la plate-forme dans les temps.

Nous avons donc les métriques suivantes :

1. Nombre d'échantillons par minute
2. Durée du test
3. Temps de réponse en ms

Nous utilisons l'outil JMeter, qui effectue des appels HTTP directement sur les serveurs, de cette façon nous ne dépendons pas des contraintes des postes clients, mais testons bien la plate-forme applicative.

Les tests de performances sont effectués manuellement à chaque livraison. Il n'y a pas de valeur ajoutée à les automatiser car ils sont longs pour être effectué à chaque modification du code. Les résultats détaillées sont archivés par la société Dream IT et les rapports agrégés sont disponibles dans ce document.

Nous avons choisi de tester les scénarios qui seront les plus sollicités dans un cadre d'utilisation réelle : affichage de la page d'accueil, connexion avec une connexion réussie et une autre incorrecte.

Les tests ont été effectués sur un serveur virtualisé ayant la configuration suivante :

- 10Go de disque

- 1Go de RAM
- 1 VCPU
- OS debian squeeze 64 bits

De plus, cette VM est également utilisée comme serveur d'intégration continue effectuant des builds réguliers et hébergeant le serveur web, le serveur applicatif et la base de données. Cela impacte donc également les résultats des tests de performances.

Cette configuration ne reflète pas de la configuration cible pour un environnement de production. La société ProxyStation ne mets pas à disposition de machine de production pour les tests de performances.

Dream IT conseille vivement un environnement de production redondé actif/passif ou actif/actif avec 4 serveurs physiques. 2 de bases de données et 2 applicatifs avec le serveur web et le serveur d'application disponibles.

## Tests de sécurité

Les exigences de sécurité n'ont pas fait partie de besoins clairement émis par le client. Néanmoins, Dream IT a défini quels tests devraient être réalisés pour être conscients des risques de sécurité de l'application. Les contrôles définis par la suite sont issues du standard OWASP, association actuellement leader en sécurité informatique.

Le code couleur suivant a été utilisé :

- gris : non applicable au projet, fonctionnalité non disponible par exemple
- non testable pour raisons techniques
- testé et validé
- testé et non validé

Les actions préalables suivantes ont été effectuées

- Manually explore the site
- Spider/crawl for missed or hidden content
- Check for files that expose content, such as robots.txt, sitemap.xml, .DS\_Store
- Check the caches of major search engines for publicly accessible sites
- Check for differences in content based on User Agent (eg, Mobile sites, access as a Search engine Crawler)
- Perform Web Application Fingerprinting
- Identify technologies used
- Identify user roles
- Identify application entry points
- Identify client-side code



- Identify multiple versions/channels (e.g. web, mobile web, mobile app, web services)
- Identify co-hosted and related applications
- Identify all hostnames and ports
- Identify third-party hosted content

## Configuration Management

- Check for commonly used application and administrative URLs : identifiées
- Check for old, backup and unreferenced files : aucune référence à d'anciens fichiers javascript ou librairies
- Check HTTP methods supported and Cross Site Tracing (XST) : utilisation outil Wapiti
- Test file extensions handling : pas de chargement de fichier dans l'applicatif
- Test for security HTTP headers (e.g. CSP, X-Frame-Options, HSTS) : utilisation outil Wapiti
- Test for policies (e.g. Flash, Silverlight, robots) : pas de briques spécifiques, standard HTML5/CSS3
- Test for non-production data in live environment, and vice-versa : données de tests non réelles et anonymes, environnement de production pas encore disponible
- Check for sensitive data in client-side code (e.g. API keys, credentials) : aucune données sensibles dans le code client

## Secure Transmission : pas de HTTPS en place

- Check SSL Version, Algorithms, Key length
- Check for Digital Certificate Validity (Duration, Signature and CN)
- Check credentials only delivered over HTTPS
- Check that the login form is delivered over HTTPS
- Check session tokens only delivered over HTTPS
- Check if HTTP Strict Transport Security (HSTS) in use

## Authentication

- Test for user enumeration : non possible
- Test for authentication bypass : non possible, protégé spring security injection
- Test for bruteforce protection : possible, pas de limite d'appels
- Test password quality rules : validation des règles de taille
- Test remember me functionality : fonctionnalité non disponible
- Test for autocomplete on password forms/input : si activé navigateur client uniquement
- Test password reset and/or recovery : non disponible , uniquement processus manuel
- Test password change process : process manuel validé
- Test CAPTCHA : fonctionnalité non disponible
- Test multi factor authentication : fonctionnalité non disponible
- Test for logout functionality presence : validé
- Test for cache management on HTTP (eg Pragma, Expires, Max-age) : validé
- Test for default logins : pas de login par défaut en place ni de standard utilisé
- Test for user-accessible authentication history : pas de stockage

- Test for out-of channel notification of account lockouts and successful password changes : pas de communication email relative aux comptes
- Test for consistent authentication across applications with shared authentication schema / SSO : pas de SSO ou authentification multi applications

### Session Management

- Establish how session management is handled in the application (eg, tokens in cookies, token in URL) : uniquement cookies
- Check session tokens for cookie flags (httpOnly and secure) : httpOnly non présent
- Check session cookie scope (path and domain) : validé suivant standard
- Check session cookie duration (expires and max-age) : timeout bien présent
- Check session termination after a maximum lifetime : validé
- Check session termination after relative timeout : timeout pas en place
- Check session termination after logout : timeout pas en place
- Test to see if users can have multiple simultaneous sessions : possible
- Test session cookies for randomness : validé 10 différents
- Confirm that new session tokens are issued on login, role change and logout : validé
- Test for consistent session management across applications with shared session management : pas de partage session entre plusieurs applications
- Test for session puzzling : non possible
- Test for CSRF and clickjacking : test wapiti

### Authorization

- Test for path traversal : non possible
- Test for bypassing authorization schema : non possible
- Test for vertical Access control problems (a.k.a. Privilege Escalation) : non possible
- Test for horizontal Access control problems (between two users at the same privilege level) : non possible
- Test for missing authorization : non identifié

### Data Validation

- Test for Reflected Cross Site Scripting : test wapiti, voir résultats
- Test for Stored Cross Site Scripting : test wapiti, voir résultats
- Test for DOM based Cross Site Scripting : test wapiti, voir résultats
- Test for Cross Site Flashing : test wapiti, voir résultats
- Test for HTML Injection : test wapiti, voir résultats
- Test for SQL Injection : test wapiti, voir résultats
- Test for LDAP Injection : pas d'utilisation LDAP
- Test for ORM Injection : pas d'outil pour tester
- Test for XML Injection : pas d'outil pour tester
- Test for XXE Injection : pas d'outil pour tester
- Test for SSI Injection : pas d'outil pour tester
- Test for XPath Injection : test wapiti, voir résultats
- Test for XQuery Injection : test wapiti, voir résultats

- Test for IMAP/SMTP Injection : pas d'utilisation serveur SMTP/IMAP dédié
- Test for Code Injection : test wapiti, voir résultats
- Test for Expression Language Injection : test wapiti, voir résultats
- Test for Command Injection : test wapiti, voir résultats
- Test for Overflow (Stack, Heap and Integer) : test wapiti, voir résultats
- Test for Format String : test wapiti, voir résultats
- Test for incubated vulnerabilities : test wapiti, voir résultats
- Test for HTTP Splitting/Smuggling : test wapiti, voir résultats
- Test for HTTP Verb Tampering : test wapiti, voir résultats
- Test for Open Redirection : test wapiti, voir résultats
- Test for Local File Inclusion : pas d'échanges de fichiers
- Test for Remote File Inclusion : pas d'échanges de fichiers
- Compare client-side and server-side validation rules : test wapiti, voir résultats
- Test for NoSQL injection : test wapiti, voir résultats
- Test for HTTP parameter pollution : test wapiti, voir résultats
- Test for auto-binding : test wapiti, voir résultats
- Test for Mass Assignment : pas d'outil pour tester
- 
- Test for NULL/Invalid Session Cookie : test wapiti, voir résultats

#### Denial of Service

- Test for anti-automation : pas de blocage par IP si trop d'appels
- Test for account lockout : validé
- Test for HTTP protocol DoS : validé
- Test for SQL wildcard DoS : validé

#### Business Logic

- Test for feature misuse : validé
- Test for lack of non-repudiation : pas d'outillage
- Test for trust relationships : pas d'outillage
- Test for integrity of data : validé, foreign key requises
- Test segregation of duties

#### Cryptography : pas de chiffrement des données

- Check if data which should be encrypted is not
- Check for wrong algorithms usage depending on context
- Check for weak algorithms usage
- Check for proper use of salting
- Check for randomness functions

#### Risky Functionality - File Uploads : pas de fichiers échangé

- Test that acceptable file types are whitelisted
- Test that file size limits, upload frequency and total file counts are defined and are enforced

- Test that file contents match the defined file type
- Test that all file uploads have Anti-Virus scanning in-place.
- Test that unsafe filenames are sanitised
- Test that uploaded files are not directly accessible within the web root
- Test that uploaded files are not served on the same hostname/port
- Test that files and other media are integrated with the authentication and authorisation schemas

#### Risky Functionality - Card Payment

- Test for known vulnerabilities and configuration issues on Web Server and Web Application : fait suivant site OWASP
- Test for default or guessable password : validé
- Test for non-production data in live environment, and vice-versa : validé
- Test for Injection vulnerabilities : testé, tout n'est pas validé
- Test for Buffer Overflows : vulnérable
- Test for Insecure Cryptographic Storage : pas de chiffrement
- Test for Insufficient Transport Layer Protection : pas de protocole HTTPS, uniquement HTTP
- Test for Improper Error Handling : validé IHM et logs serveurs
- Test for all vulnerabilities with a CVSS v2 score > 4.0 : CSS3 utilisé
- Test for Authentication and Authorization issues : validé
- Test for CSRF : testé

#### HTML 5

- Test Web Messaging : pas de messagerie instantanée disponible
- Test for Web Storage SQL injection : testé
- Check CORS implementation : fonctionnalité non disponible
- Check Offline Web Application : fonctionnalité non disponible

## Tests exploratoires

Des tests exploratoires ont été planifiés dans le projet afin de trouver d'éventuels bugs ou améliorations qui n'auraient pas été anticipés dans les spécifications. Le but est de se mettre à la place d'un nouvel utilisateur et d'explorer les différents cheminements de pages possibles.

## Périmètre couvert

Nous couvrons toutes les couches logicielles de tests en assurant les tests unitaires en isolation et les tests d'intégration. Les tests unitaires ne sont pas effectués sur la couche contrôleur de la partie MVC car nous jugeons que les tests d'intégration suffisent, et que la simulation d'un environnement web (requêtes etc) est trop lourde à maintenir.

## Revue de code

Les revues de code nous servent à partager la connaissance au sein de l'équipe, et également de réduire la dette technique sur le projet.

Chaque membre de l'équipe présente chaque semaine le code qu'il a développé de façon globale, en parcourant le code, afin de donner les clés des fonctionnalités aux autres membres. Cela permet de diffuser la connaissance et de faciliter l'évolution d'une fonctionnalité par différents développeurs.

Toutes les semaines, le responsable qualité édite une liste des correctifs à effectuer dans les 2 jours qui suivent. Le but de cette réunion est d'éviter d'accumuler trop de dette technique qui coûte cher sur le long terme. En effet, l'accumulation de petits défauts conduit sur le long terme à des problèmes de maintenance dus à de mauvais choix. Les correctifs peuvent concerner du nommage de méthode, de variable, des problèmes de conception. L'outil sonar nous aide à détecter les erreurs en cas de code non conforme aux standards du marché.

## Améliorations

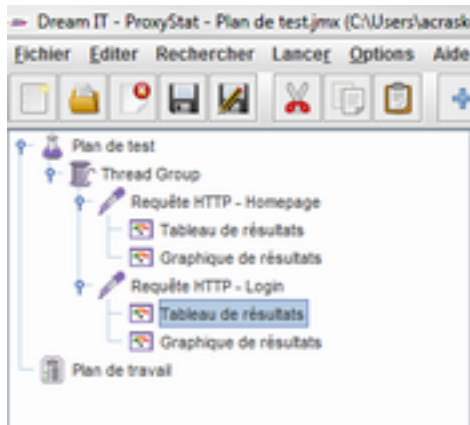
En 1 mois nous n'avons pas eu le temps d'effectuer tous nos objectifs mais avons les plans d'actions adéquats. Nous allons viser le taux de couverture de tests unitaires pour la V2.

## Configuration

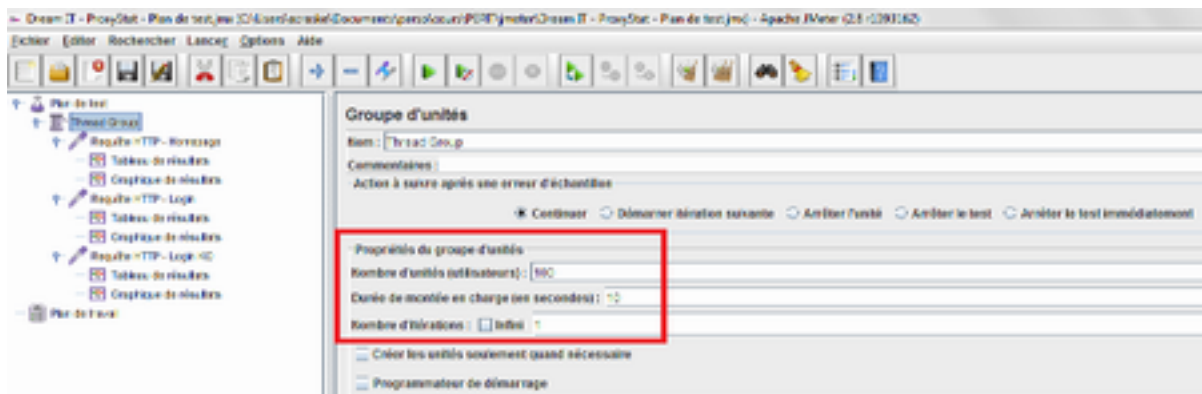
## Tests de performances

L'outil JMeter a été choisi car éprouvé dans le cadre de tests de performances. C'est un logiciel open-source client lourd Java. La version utilisée est la 2.8, dernière version stable à cette date.

Les 3 cas de tests ont été configurés en respectant le standard de configuration JMeter : 1 thread group puis une déclinaison des cas de tests.



La configuration des paramètres de chargement de la plateforme se configurent au niveau du Thread group. Ces derniers sont changés à chaque nouvelle campagne de test en fonction de la charge cible.

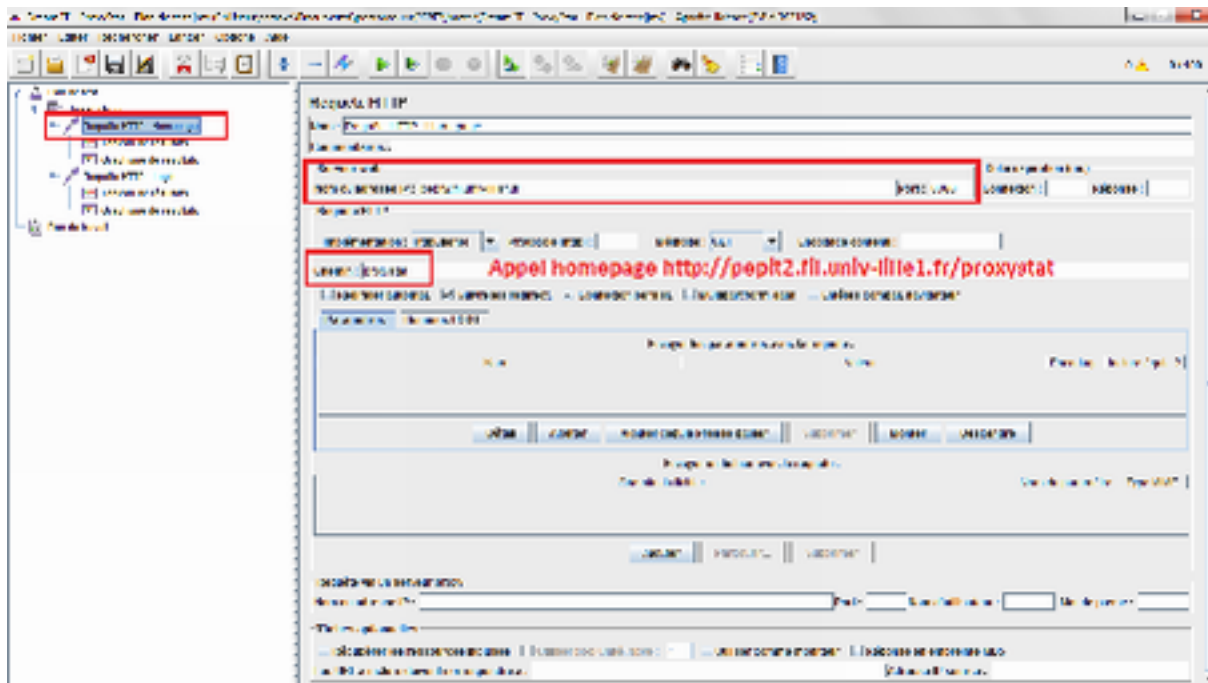


Chacun des scénarios de tests est un appel HTTP. On utilise un tableau stockant tous les résultats qui seront archivés, un tableau agrégeant les résultats qui sont présents dans ce document et le graphique qui permet de détecter rapidement les anomalies.

Chacun des scénarios est décrit dans les paragraphes suivants.

## Scénario page d'accueil

Ce scénario fait un appel HTTP simple sur l'URL de la page d'accueil. Le but est d'assurer un affichage de la page d'accueil dans des temps de réponses acceptables pour l'utilisateur afin qu'il reste sur le site. La configuration est présentée ci-dessous.

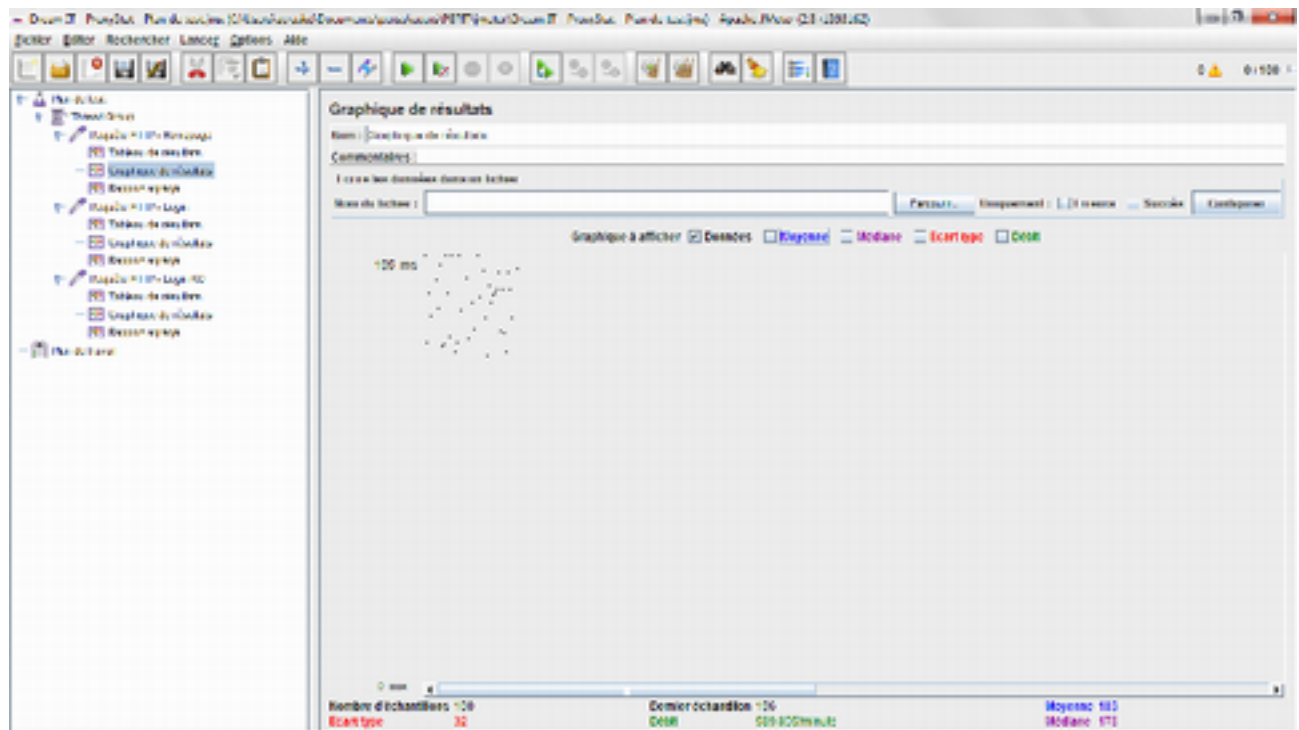


Pour rendre les choses plus concrète les différents types de résultats sont présentés ci-après uniquement pour ce scénario.

Le tableau complet de résultat :



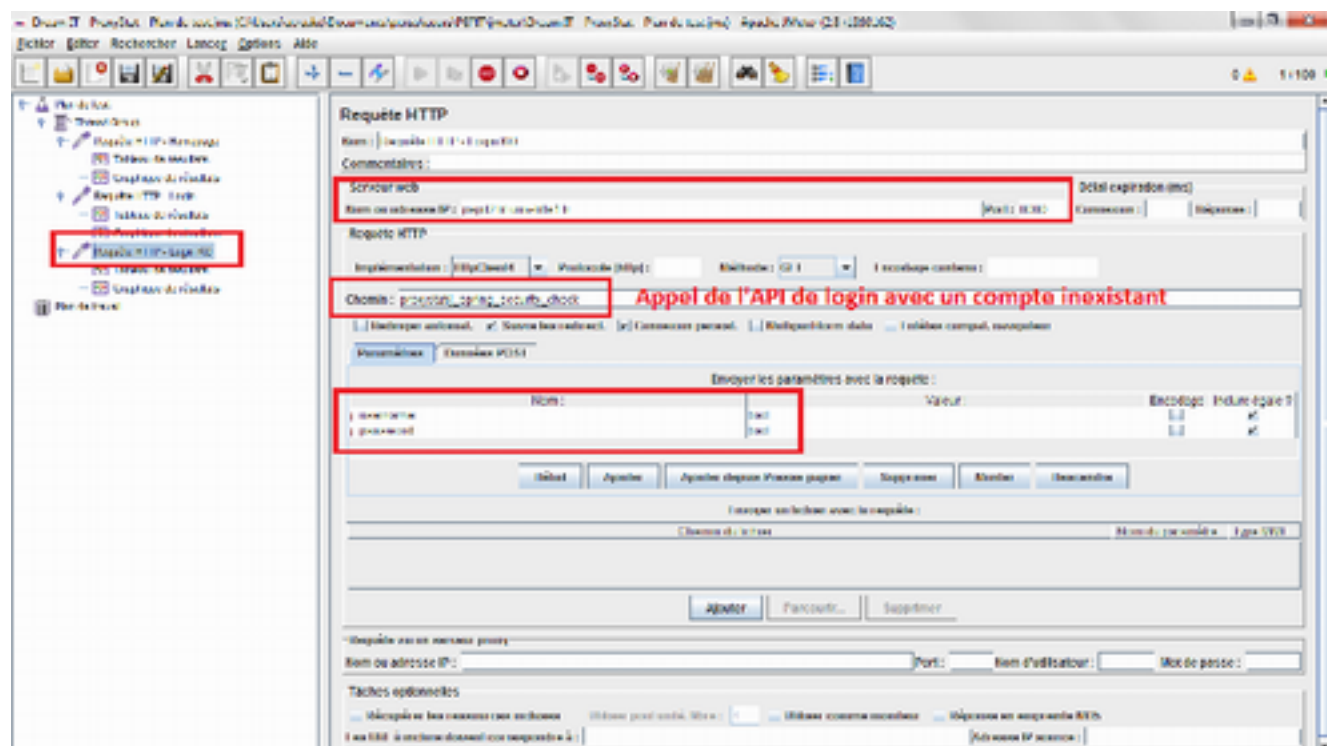




## Scénario authentification valide

Ce scénario fait un appel HTTP GET sur l'URL d'authentification avec des identifiants existants en base de données. Le but est de valider qu'on pourra authentifier les clients dans des temps de réponses acceptables pour qu'ils restent sur le site et achètent potentiellement.





# **Enregistrement des résultats**

**V1 : 22 février 2013**

Tests unitaires

Tests d'intégration

## Tests de performances

Durée : 30 minutes

Nombre d'utilisateurs : 1000

Résultats :

Scénario	Temps réponse moyen (< 1s ?)	% Erreur	Résultat validé / invalidé
Homepage	779 ms	0,30%	Validé
Login OK	488 ms	0,10%	Validé
Login KO	351 ms	0%	Validé

### Homepage

Libellé	# Echantillons	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête HTTP - Homepage	1000	779	266	2072	143	16407	0,30%	33,3/min	2,7
TOTAL	1000	779	266	2072	143	16407	0,30%	33,3/min	2,7

### Login OK

Libellé	# Echantillons	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête HTTP -	1000	488	194	901	100	17780	0,10%	33,3/min	2,8
TOTAL	1000	488	194	901	100	17780	0,10%	33,3/min	2,8

### Login KO

Libellé	# Echantillons	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête HTTP -	1000	351	160	569	99	17400	0,00%	33,3/min	2,8
TOTAL	1000	351	160	569	99	17400	0,00%	33,3/min	2,8

Durée : 30 minutes

Nombre d'utilisateurs : 5000

Résultats :

Scénario	Temps réponse moyen (< 1s ?)	% Erreur	Résultat validé / invalidé
Homepage	362	0 %	Validé
Login OK	176	0 %	Validé
Login KO	154	0 %	Validé

Homepage

Libellé	# Echantillons	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête HTTP - Homepage	3975	362	192	456	139	15337	0.00%	2.8/sec	13.7
TOTAL	3975	362	192	456	139	15337	0.00%	2.8/sec	13.7

Login OK

Libellé	# Echantillons	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête HTTP - ...	3975	174	141	222	97	6475	0.00%	2.8/sec	14.1
TOTAL	3975	174	141	222	97	6475	0.00%	2.8/sec	14.1

Login KO

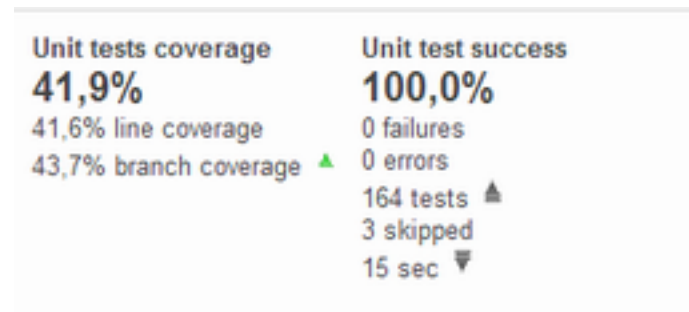
Libellé	# Echantillons	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête HTTP - ...	3975	154	107	205	97	12370	0.00%	2.8/sec	14.1
TOTAL	3975	154	107	205	97	12370	0.00%	2.8/sec	14.1

## V2 : 26 mars 2013

### Tests unitaires
















Pour les tests unitaires de la version 2, nous atteignons une couverture de 41,9%.

Nous pouvons remarquer que par rapport à la première livraison, le taux de couverture a bien augmenté.



Depuis la mise en place de l'intégration continue avec jenkins, nous pouvons assurer que le code commité n'entraîne pas de régression au niveau du code déjà existant.

La vue par composants de sonar présentée ci-dessous nous montre pour chaque ‘package composant’ du projet les taux de couverture de code :’

Package	Rate compliance	Coverage	Build time	Links
 <a href="#">project</a>	0.0%	41,5%	00:04	
+ Package				
 <a href="#">com.dream1.project.integration</a>			00:04	
 <a href="#">com.dream1.project.model.controller</a>	0.0%	1.0%	00:04	
 <a href="#">com.dream1.project.model.dao</a>	100.0%		00:04	
 <a href="#">com.dream1.project.model.dao.impl</a>	100.0%	71.9%	00:04	
 <a href="#">com.dream1.project.model.dao.gifts</a>	0.0%	62.1%	00:04	
 <a href="#">com.dream1.project.model.dto</a>	100.0%	61.0%	00:04	
 <a href="#">com.dream1.project.model.function</a>	0.0%	33.4%	00:04	
 <a href="#">com.dream1.project.model.immutable</a>	100.0%	0.0%	00:04	
 <a href="#">com.dream1.project.model.exception</a>	100.0%	62.0%	00:04	
 <a href="#">com.dream1.project.model.factory</a>	100.0%		00:04	
 <a href="#">com.dream1.project.model.factory.impl</a>	100.0%	100.0%	00:04	
 <a href="#">com.dream1.project.model.service</a>	0.0%		00:04	
 <a href="#">com.dream1.project.model.service.impl</a>	0.0%	73.7%	00:04	
 <a href="#">com.dream1.project.view</a>	100.0%	0.0%	00:04	

Nous pouvons remarqué que par rapport à ce qui a été convenu, et par rapport à nos engagements, les différents taux de couverture de code ont bien été respectés.

Notamment au niveau des couches de service, nous avons convenu un minimum de 80%, qui ont bien été respecté étant donné actuellement, nous avons un taux de 94,5%.

Ci-dessous un tableau récapitulatif de ce qui a été convenu et de l'état actuel des taux de couverture de code par composant :

Package	Taux de couverture requis	Taux de couverture actuel
<b>Model</b>	0%	<b>0%</b>
<b>Dao</b>	100%	<b>100%</b>
<b>Controlleur</b>	Intégration	<b>Intégration</b>
<b>Vue</b>	Intégration	<b>Intégration</b>
<b>Service</b>	80%	<b>94,5%</b>



## Tests d'intégration

Comme évoqué, les tests d'intégration ont été réalisés avec selenium IDE, et ont été ensuite convertis en tests junit, en utilisant le webdriver de selenium.

La partie vue et la partie contrôleur ont été testées par intégration.

Nous pouvons noter que le taux de couverture pour les contrôleurs est de 94% :

4	 com.streml.personal.model.controllers	94,0%	1,3%	00:04
---	---	-------	------	-------

Et que la partie vue quand à elle a été couverte à 100%

4	 com.streml.personal.view	100,0%	0,0%	00:04
---	--	--------	------	-------

## Tests de performances

Durée : 30 minutes

Nombre d'utilisateurs : 1000

Résultats :

Scénario	Temps réponse moyen (< 1s ?)	% Erreur	Résultat validé / invalidé
Homepage	153 ms	0 %	Validé
Login OK	81 ms	0 %	Validé
Login KO	80 ms	0 %	Validé

### Homepage

Libellé	# Echantillo...	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête H...	1000	153	114	125	106	3226	0,00%	33,4/min	3,7
TOTAL	1000	153	114	125	106	3226	0,00%	33,4/min	3,7

### Login OK

Libellé	# Echantillo...	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête H...	1000	81	85	90	56	797	0,00%	33,4/min	3,8
TOTAL	1000	81	85	90	56	797	0,00%	33,4/min	3,8

### Login KO

Libellé	# Echantillo...	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête H...	1000	80	83	89	56	757	0,00%	33,4/min	3,8
TOTAL	1000	80	83	89	56	757	0,00%	33,4/min	3,8

Durée : 30 minutes

Nombre d'utilisateurs : (5000) 3137

Résultats :

Scénario	Temps réponse moyen (< 1s ?)	% Erreur	Résultat validé / invalidé
Homepage	113 ms	0 %	Validé
Login OK	80 ms	0 %	Validé

Login KO	78 ms	0 %	Validé
----------	-------	-----	--------

Malgré l'ajout d'une charte graphique avec de multiples fichiers javascript, css et images, les temps de réponses restent stables.

#### Homepage

Libellé	# Echantillo...	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête H...	3137	134	113	144	106	3220	0,00%	2.8/sec	18,5
TOTAL	3137	134	113	144	106	3220	0,00%	2.8/sec	18,5

#### Login OK

Libellé	# Echantillo...	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête H...	3137	80	85	90	55	822	0,00%	2.8/sec	18,9
TOTAL	3137	80	85	90	55	822	0,00%	2.8/sec	18,9

#### Login KO

Libellé	# Echantillo...	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête H...	3137	78	84	89	55	556	0,00%	2.8/sec	18,9
TOTAL	3137	78	84	89	55	556	0,00%	2.8/sec	18,9

## V3 : 7 juin 2013































### Tests unitaires

Pour les tests unitaires de la version 3, nous atteignons une couverture de 47%.

Nous pouvons remarquer que par rapport à la première et la deuxième livraison, le taux de couverture a bien augmenté.

Unit tests coverage	Unit test success
<b>47,0%</b>	<b>100,0%</b>
47,1% line coverage	0 failures
46,7% branch coverage	0 errors
	308 tests ▲
	32.4 sec ▲

La vue par composants de sonar présentée ci-dessous nous montre pour chaque package composant' du projet les taux de couverture de code :'

Name	Rules compliance	Coverage	Build time	Links
  <b>ecmsuite</b>	100.0%	47.0%	20:04	
+ Name				
  <b>com.damit.ecmsuite.integration</b>			20:04	
  <b>com.damit.ecmsuite.model.controllers</b>	100.0%	0.0%	20:04	
  <b>com.damit.ecmsuite.model.dbo</b>	100.0%		20:04	
  <b>com.damit.ecmsuite.model.dbo.maj</b>	100.0%	59.0%	20:04	
  <b>com.damit.ecmsuite.model.dbo.parten</b>	100.0%	52.0%	20:04	
  <b>com.damit.ecmsuite.model.dbo</b>	100.0%	60.0%	20:04	
  <b>com.damit.ecmsuite.model.entites</b>	99.0%	44.0%	20:04	
  <b>com.damit.ecmsuite.model.enumeration</b>	100.0%	0.0%	20:04	
  <b>com.damit.ecmsuite.model.exceptions</b>	100.0%	90.0%	20:04	
  <b>com.damit.ecmsuite.model.factories</b>	100.0%		20:04	
  <b>com.damit.ecmsuite.model.factories.maj</b>	100.0%	100.0%	20:04	
  <b>com.damit.ecmsuite.model.service</b>	99.0%		20:04	
  <b>com.damit.ecmsuite.model.service.impl</b>	99.0%	76.2%	20:04	
  <b>com.damit.ecmsuite.util</b>	100.0%	0.0%	20:04	

Nous pouvons remarqué que le taux de couverture de code des services est de 76,2%.

Pour cette troisième version, il manque 4% de code non couvert par rapport aux objectifs définies de 80%.

Cela se justifie par le fait que pour le module de création de compte, il n'est pas nécessaire de passer par tous les cas possibles (tester des champs vides), étant donné que cela a été testé en intégration. Il nous a donc parut plus sensé de tester ce module en intégration que faire des cas de tests triviaux seulement pour atteindre une couverture de code élevée.

Nous pouvons d'ailleurs remarquer dans la partie suivant de ce document que le taux de couverture de code pour les controleurs a nettement augmenté par rapport à la deuxième version du projet.

Package	Taux de couverture requis	Taux de couverture actuel
---------	---------------------------	---------------------------

<b>Model</b>	0%	0%
<b>Dao</b>	100%	100%
<b>Controlleur</b>	Intégration	<b>Intégration</b>
<b>Vue</b>	Intégration	<b>Intégration</b>
<b>Service</b>	80%	76,2%.

## Tests d'intégration

Comme évoqué, les tests d'intégration ont été réalisés avec selenium IDE, et ont été ensuite convertis en tests junit, en utilisant le webdriver de selenium.

La partie vue et la partie contrôleur ont été testées par intégration.

Nous pouvons noter que le taux de couverture pour les contrôleurs est de 100% contre 94% lors de la deuxième version du projet.

Cette augmentation se justifie par le fait qu'il a été jugé préférable de faire des tests d'intégration complets pour certains modules que de faire des tests unitaires triviaux.

  <a href="#">com.dreamit.proxystat.model.controllers</a>	100,0%	0,9%
---	--------	------

La partie vue quand à elle a aussi été couverte à 100% en intégration :

  <a href="#">com.dreamit.proxystat.view</a>	100,0%	0,0%
--	--------	------

## Tests de performances

Durée : 30 minutes

Nombre d'utilisateurs : 1000

Résultats :

Scénario	Temps réponse moyen (< 1s ?)	% Erreur	Résultat valide / invalidé
Homepage	3526 ms en raison du timeout google analytics, sinon 337 ms	0,30 %	Validé
Login OK	573 ms	0,20 %	Validé
Login KO	731 ms	0 %	Validé

### Homepage

Libellé	# Echantillo...	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête H...	1000	3526	337	1423	188	153420	0,30%	33,4/min	4,3
TOTAL	1000	3526	337	1423	188	153420	0,30%	33,4/min	4,3

### Login OK

Libellé	# Echantillo...	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête H...	1000	573	227	634	105	134092	0,20%	33,4/min	4,4
TOTAL	1000	573	227	634	105	134092	0,20%	33,4/min	4,4

### Login KO

Libellé	# Echantillo...	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête H...	1000	731	210	554	108	140138	0,00%	33,4/min	4,4
TOTAL	1000	731	210	554	108	140138	0,00%	33,4/min	4,4

Durée : 30 minutes

Nombre d'utilisateurs : 5000 (3162)

Résultats :

Scénario	Temps réponse moyen (< 1s ?)	% Erreur	Résultat valide / invalidé
Homepage	2565ms en raison du timeout google analytics, sinon 340 ms	0,25 %	Validé
Login OK	538 ms	0,22 %	Validé
Login KO	646 ms	0,03 %	Validé

Malgré l'ajout d'une charte graphique avec de multiples fichiers javascript, css et images, les temps de réponses restent stables.

#### Homepage

Libellé	# Echantillons	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête HTTP ...	3162	2565	646	1405	180	105691	0.25%	1,4/sec	10.8
TOTAL	3162	2565	646	1405	180	105691	0.25%	1,4/sec	10.8

#### Login OK

Libellé	# Echantillons	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête HTTP ...	3162	538	381	876	101	101067	0.22%	1,4/sec	11.0
TOTAL	3162	538	381	876	101	101067	0.22%	1,4/sec	11.0

#### Login KO

Libellé	# Echantillons	Moyenne	Médiane	90e centile	Min	Max	% Erreur	Débit	Ko/sec
Requête HTTP ...	3162	646	393	858	101	100651	0.03%	1,4/sec	11.0
TOTAL	3162	646	393	858	101	100651	0.03%	1,4/sec	11.0



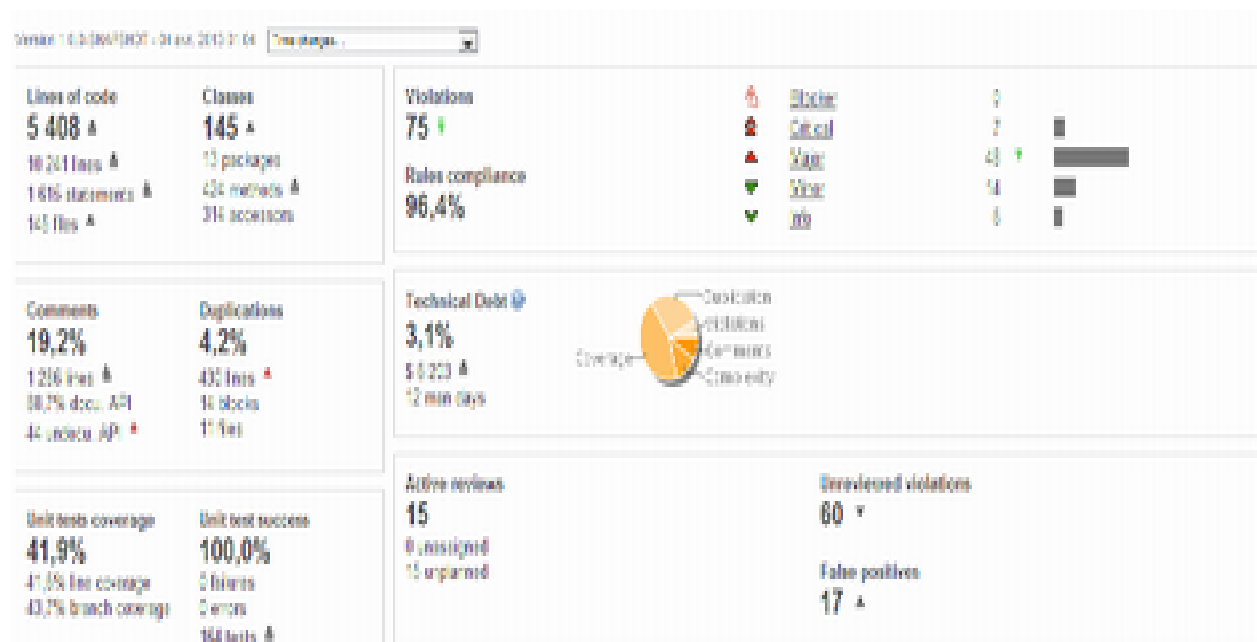
## Mesures avec l'outil Sonar

Ci-dessous le tableau de bord de notre projet sous Sonar. Toutes les alertes ne sont pas traitées mais nous nous sommes focalisé sur les bloquantes et critiques. Le responsable qualité définira les actions les plus importantes à mener pour les réduire.

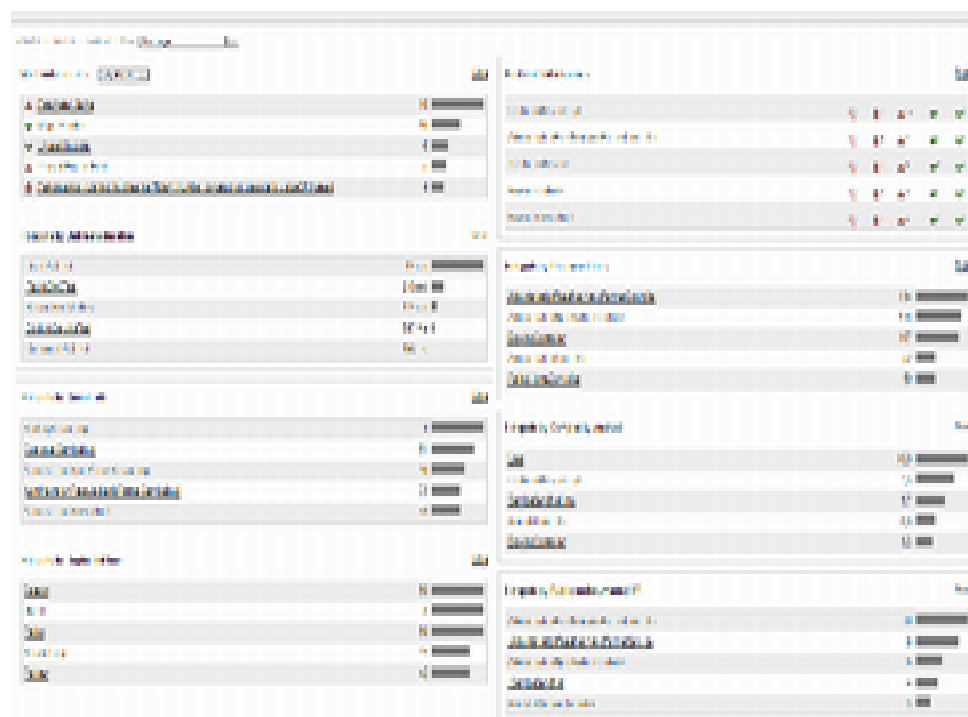
Nous atteignons bien les 100% de succès des tests unitaires.

Nous avons 4% de fausses-alertes sur la duplication dû à des problèmes d'analyse de l'outil. Preuve qu'il faut toujours investiguer le calcul d'une mesure avant de s'y fier.

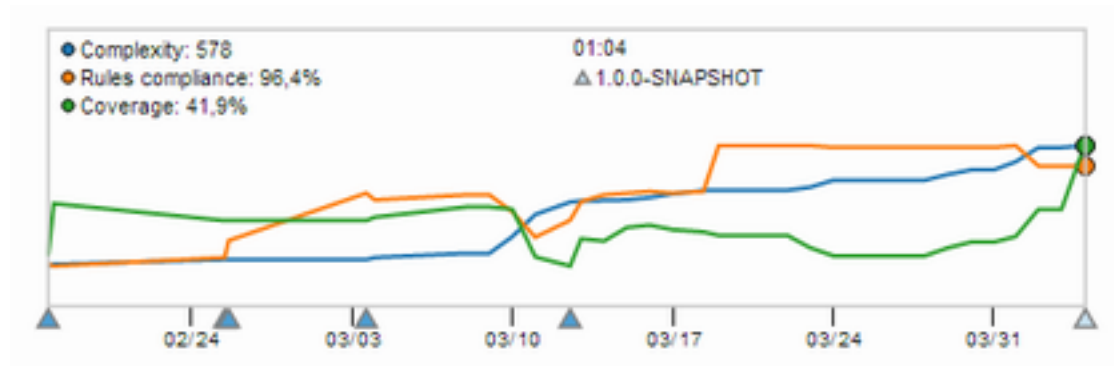
Le bon point est la documentation du code, nous faisons en effet le choix de documenter au maximum le code, ce qui est utile aux développeurs au lieu de le faire dans un document séparé qui est vite oublié ou perdu.



On remarque que la majeure partie des erreurs sont dues à des problèmes de visibilité des champs, liées à des contraintes du framework de mapping objet/relationnel.



On note bien une diminution du nombre d'erreurs mais une augmentation de la complexité. Nous devons donc continuer notre plan d'action de revue de code et effectuer également une revue d'architecture sur les classes les plus complexes.



Nous avons atteints nos objectifs ;  
les 100% de couverture de code sur la couche DAO sont atteints (dao.impl) et 94,5% sur la couche service par rapport à notre objectif de 80%.  
Nous montrons notre capacité à suivre les règles établies en début de projet.

Name	Rules compliance	Coverage	Build time	Links
<a href="#">dao</a>	96,4%	41,9%	01:04	
+ Name				
<a href="#">com.dreamt.perystal.integration</a>			01:04	
<a href="#">com.dreamt.perystal.model.entities</a>	94,6%	1,3%	01:04	
<a href="#">com.dreamt.perystal.model.dao</a>	90,6%		01:04	
<a href="#">com.dreamt.perystal.model.dao.impl</a>	90,6%	73,3%	01:04	
<a href="#">com.dreamt.perystal.model.dao.pattern</a>	90,1%	52,6%	01:04	
<a href="#">com.dreamt.perystal.model.dto</a>	90,6%	53,3%	01:04	
<a href="#">com.dreamt.perystal.model.entities</a>	90,6%	33,6%	01:04	
<a href="#">com.dreamt.perystal.model.enumeration</a>	90,6%	0,0%	01:04	
<a href="#">com.dreamt.perystal.model.exceptions</a>	90,6%	50,0%	01:04	
<a href="#">com.dreamt.perystal.model.factory</a>	90,6%		01:04	
<a href="#">com.dreamt.perystal.model.factory.impl</a>	90,6%	100,0%	01:04	
<a href="#">com.dreamt.perystal.model.service</a>	84,6%		01:04	
<a href="#">com.dreamt.perystal.model.service.impl</a>	94,6%	73,3%	01:04	
<a href="#">com.dreamt.perystal.view</a>	90,6%	0,0%	01:04	

[Help](#)

Dependency
Suspect dependency (cycle)
- uses >
- uses >

com.dreamit.proxystat.model.controllers	-												
com.dreamit.proxystat.model.dao.impl		-											
com.dreamit.proxystat.model.enumeration	1		-										
com.dreamit.proxystat.model.factory.impl				-									
com.dreamit.proxystat.model.service.impl					-								
com.dreamit.proxystat.view						-							
com.dreamit.proxystat.model.dao	1	32			32		-						
com.dreamit.proxystat.model.factory	7			7				-					
com.dreamit.proxystat.model.service	13				9				-				
com.dreamit.proxystat.model.dto	1	2		1	4		2	1	4	-			
com.dreamit.proxystat.model.entities	17	37		18	35		34	18	28	9	-		
com.dreamit.proxystat.model.dao.pattern		32					32					-	
com.dreamit.proxystat.model.exceptions	2				6				6				-

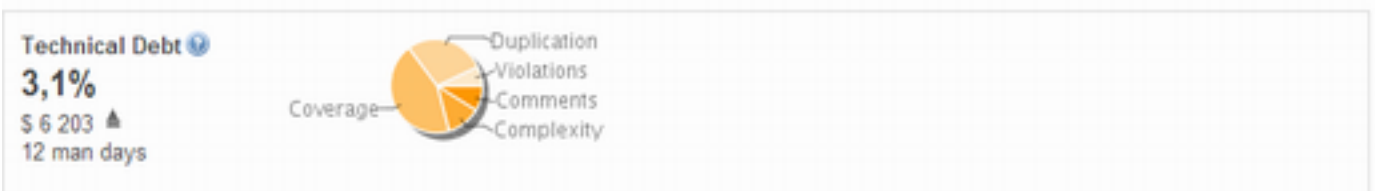
[illegible]

Enfin, des 'modules' ont été ajouté à sonar pour permettre d'améliorer la qualité du code fournie.

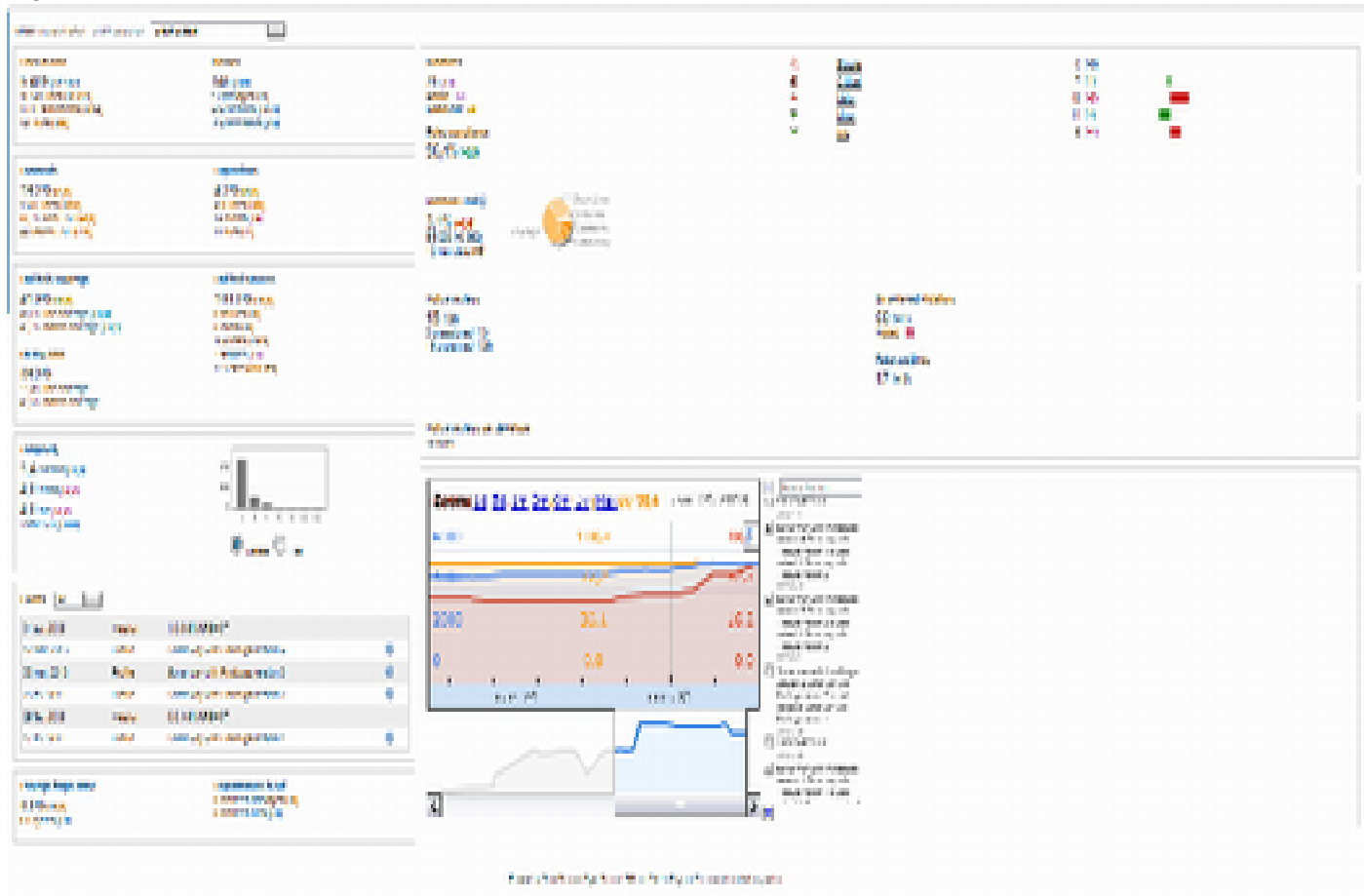
Notamment, nous pouvons désormais assigner des tâches de refactoring ou de corrections de buggs potentiels à chaque membre de l'équipe, ce qui permet d'avoir un meilleur retour et un réel investissement pour l'amélioration de la qualité du code :

Active reviews per developer		
<u>rud</u> y	6	<div></div>
<u>eric</u>	5	<div></div>
<u>tan</u> k	4	<div></div>

Grâce à cela, nous avons donc pu réduire nos dettes techniques pour le projet proxystat au cours de ces 2 dernier mois, et nous allons continuer à réduire cette dernière .



Pour conclure, vous pouvez trouver ci-dessous un récapitulatif des différences de la qualité du logiciel fourni par rapport à la première livraison :



## Tests multi navigateur

Les tests multi navigateur permettent de tester que l'application est compatible sur plusieurs navigateurs.

Nous avons pris en compte notamment les 3 navigateurs principaux les plus utilisés, à savoir : Google Chrome, Mozilla Firefox et Internet Explorer.

Les fonctionnalités pris en compte lors du test multi navigateur sont les fonctionnalités principales de l'application, à savoir : le passage de commande, la création de compte et l'interface pour les statistiques.

Ci-dessous les détails concernant les tests :

	Google chrome	Mozilla Firefox	Internet Explorer
Passage de commande	OK	- L'ascenseur pour le choix du nombre de personne et du prix n'est pas rendu visuellement, mais on peut cependant rentrer un nombre sans problème	- L'ascenseur pour le choix du nombre de personne et du prix n'est pas rendu visuellement, mais on peut cependant rentrer un nombre sans problème
Création de compte	OK	OK	OK
Reporting/Statistiques	OK	OK	OK

## Tests de sécurité

Seuls les tests automatisés ont été enregistrés ici pour leur facilité de traçabilité. Les tests manuels sont uniquement répertoriés dans le chapitre relatif aux tests de sécurité.

### Résultats des tests automatisés

Le tableau suivant est disponible en annexe dans l'archive `security-tests-results_no-auth.tar.gz`.

	SQL Injection (2)	Blind SQL Injection (2)	File Handling (3)	Cross Site Scripting (4)	CRLF (6)	Commands execution (6)	Resource consumption (7)	Access Bypass (8)	Backup file (9)	Potentially dangerous file (10)
High	0	1	1	3	0	1	0	0	0	0
Medium	0	0	0	0	0	0	0	0	0	0
Low	0	0	0	0	0	0	0	0	0	0

L'outil remonte une vulnérabilité aux injections SQL qui n'a pas pu être concrétisée avec des injections de type ``OR 1=1` car nous utilisons un outil d'Object Relational Mapping (ORM) ne permettant pas les injections de ce type. C'est donc un faux positif.

Les injections SQL aveugles sont possible car il n'y a pas de filtre sur le contenu des formulaires. L'implémentation d'un filtre pour tous les champs de formulaires reçus ou l'utilisation d'annotations spécifique Spring nous permettrait de combler ce manque en validant les charges de maintenance.

Les failles de Cross-Site Scripting (XSS) remontées ont été testées manuellement par la suite avec une inclusion de `"guest<script>alert('attacked')</script>"` dans le champ mentionné mais cela n'a pas eu l'effet de faire apparaître une popup comme attendu. La faille n'est donc pas validée.

### Logs de l'outil Wapiti

Les logs ont accédé aux URLs ne nécessitant pas d'authentification afin de simuler une attaque ne nécessitant pas de connaissance particulière de l'application.

```
wapiti http://192.168.0.29:8080/proxystat/ -n 10 -b folder -u -v 1 -f html -o output
Wapiti-2.2.1 (wapiti.sourceforge.net)
```

```
.....
Notice
=====
```



This scan has been saved in the file /root/.wapiti/scans/192.168.0.29:8080.xml

You can use it to perform attacks without scanning again the web site with the "-k" parameter

[\*] Loading modules :

mod\_crlf, mod\_exec, mod\_file, mod\_sql, mod\_xss, mod\_backup, mod\_htaccess,  
mod\_blindsqli, mod\_permanentxss, mod\_nikto

Problem with local nikto database.

Downloading from the web...

[+] Launching module crlf

+ attackGET http://192.168.0.29:8080/proxystat/showAllRatings.rate  
+ attackGET http://192.168.0.29:8080/proxystat/postRating.rate  
+ attackGET http://192.168.0.29:8080/proxystat/sendMail.contact  
+ attackGET http://192.168.0.29:8080/proxystat/creation.customer  
+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=en  
[u'locale=en']  
+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=fr  
[u'locale=fr']  
+ attackGET http://192.168.0.29:8080/proxystat/j\_spring\_security\_check  
+ attackGET http://192.168.0.29:8080/proxystat/create.customer  
+ attackGET http://192.168.0.29:8080/proxystat/showFirstRatings.rate  
+ attackGET http://192.168.0.29:8080/proxystat/.  
+ attackGET http://192.168.0.29:8080/proxystat/  
+ attackGET http://192.168.0.29:8080/proxystat/returnViewContact.contact

[+] Launching module exec

+ attackGET http://192.168.0.29:8080/proxystat/showAllRatings.rate  
+ attackGET http://192.168.0.29:8080/proxystat/postRating.rate  
+ attackGET http://192.168.0.29:8080/proxystat/sendMail.contact  
+ attackGET http://192.168.0.29:8080/proxystat/creation.customer  
+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=en  
[u'locale=en']

500 HTTP Error code with

Evil url: http://192.168.0.29:8080/proxystat/home.security?locale=a%3Benv  
+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=fr  
[u'locale=fr']  
+ attackGET http://192.168.0.29:8080/proxystat/j\_spring\_security\_check  
+ attackGET http://192.168.0.29:8080/proxystat/create.customer  
+ attackGET http://192.168.0.29:8080/proxystat/showFirstRatings.rate  
+ attackGET http://192.168.0.29:8080/proxystat/.  
+ attackGET http://192.168.0.29:8080/proxystat/  
+ attackGET http://192.168.0.29:8080/proxystat/returnViewContact.contact

[+] Launching module file

+ attackGET http://192.168.0.29:8080/proxystat/showAllRatings.rate

+ attackGET http://192.168.0.29:8080/proxystat/postRating.rate  
+ attackGET http://192.168.0.29:8080/proxystat/sendMail.contact  
+ attackGET http://192.168.0.29:8080/proxystat/creation.customer  
+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=en  
[u'locale=en']  
500 HTTP Error code with

Evil url: http://192.168.0.29:8080/proxystat/home.security?  
locale=http%3A%2F%2Fwww.google.fr%2F

+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=fr  
[u'locale=fr']  
+ attackGET http://192.168.0.29:8080/proxystat/j\_spring\_security\_check  
+ attackGET http://192.168.0.29:8080/proxystat/create.customer  
+ attackGET http://192.168.0.29:8080/proxystat/showFirstsRatings.rate  
+ attackGET http://192.168.0.29:8080/proxystat/.  
+ attackGET http://192.168.0.29:8080/proxystat/  
+ attackGET http://192.168.0.29:8080/proxystat/returnViewContact.contact

[+] Launching module sql

+ attackGET http://192.168.0.29:8080/proxystat/showAllRatings.rate  
+ attackGET http://192.168.0.29:8080/proxystat/postRating.rate  
+ attackGET http://192.168.0.29:8080/proxystat/sendMail.contact  
+ attackGET http://192.168.0.29:8080/proxystat/creation.customer  
+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=en  
[u'locale=en']  
500 HTTP Error code with

Evil url: http://192.168.0.29:8080/proxystat/home.security?locale=%BF%27%22%28

+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=fr  
[u'locale=fr']  
+ attackGET http://192.168.0.29:8080/proxystat/j\_spring\_security\_check  
+ attackGET http://192.168.0.29:8080/proxystat/create.customer  
+ attackGET http://192.168.0.29:8080/proxystat/showFirstsRatings.rate  
+ attackGET http://192.168.0.29:8080/proxystat/.  
+ attackGET http://192.168.0.29:8080/proxystat/  
+ attackGET http://192.168.0.29:8080/proxystat/returnViewContact.contact

[+] Launching module xss

+ attackGET http://192.168.0.29:8080/proxystat/showAllRatings.rate  
+ attackGET http://192.168.0.29:8080/proxystat/postRating.rate  
+ attackGET http://192.168.0.29:8080/proxystat/sendMail.contact  
+ attackGET http://192.168.0.29:8080/proxystat/creation.customer  
+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=en  
[u'locale=en']  
+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=fr

[u'locale=fr']  
+ attackGET http://192.168.0.29:8080/proxystat/j\_spring\_security\_check  
+ attackGET http://192.168.0.29:8080/proxystat/create.customer  
+ attackGET http://192.168.0.29:8080/proxystat/showFirstsRatings.rate  
+ attackGET http://192.168.0.29:8080/proxystat/.  
+ attackGET http://192.168.0.29:8080/proxystat/  
+ attackGET http://192.168.0.29:8080/proxystat/returnViewContact.contact  
Found XSS in http://192.168.0.29:8080/proxystat/create.customer

with params =  
j\_adresse\_cp=r73ry88hoi&j\_adresse\_rue=lv2eeb0p0u&j\_adresse\_pays=on&j\_password=on&j\_passwordC=on&j\_surname=%22%3E%3C%2Finput%3E%3Cscript%3Ealert%28%27sr0kcsi9c6%27%29%3C%2Fscript%3E&j\_email=on&j\_name=on&j\_adresse\_ville=on  
coming from http://192.168.0.29:8080/proxystat/creation.customer  
Found XSS in http://192.168.0.29:8080/proxystat/create.customer

with params =  
j\_adresse\_cp=r73ry88hoi&j\_adresse\_rue=lv2eeb0p0u&j\_adresse\_pays=3f98jt5zfm&j\_password=gb13c75ne8&j\_passwordC=ut8kgs7mwf&j\_surname=sr0kcsi9c6&j\_email=%22%3E%3C%2Finput%3E%3Cscript%3Ealert%28%27j3cj9y4jh%27%29%3C%2Fscript%3E&j\_name=on&j\_adresse\_ville=on  
coming from http://192.168.0.29:8080/proxystat/creation.customer  
Found XSS in http://192.168.0.29:8080/proxystat/create.customer

with params =  
j\_adresse\_cp=r73ry88hoi&j\_adresse\_rue=lv2eeb0p0u&j\_adresse\_pays=3f98jt5zfm&j\_password=gb13c75ne8&j\_passwordC=ut8kgs7mwf&j\_surname=sr0kcsi9c6&j\_email=j3cj9y4jh&j\_name=%22%3E%3C%2Finput%3E%3Cscript%3Ealert%28%27bce249ierj%27%29%3C%2Fscript%3E&j\_adresse\_ville=on  
coming from http://192.168.0.29:8080/proxystat/creation.customer

[+] Launching module blindsqli  
+ attackGET http://192.168.0.29:8080/proxystat/showAllRatings.rate  
+ attackGET http://192.168.0.29:8080/proxystat/postRating.rate  
+ attackGET http://192.168.0.29:8080/proxystat/sendMail.contact  
+ attackGET http://192.168.0.29:8080/proxystat/creation.customer  
+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=en  
[u'locale=en']  
500 HTTP Error code with  
Evil url: http://192.168.0.29:8080/proxystat/home.security?locale=sleep%28%27%29%23  
+ attackGET http://192.168.0.29:8080/proxystat/home.security?locale=fr  
[u'locale=fr']  
+ attackGET http://192.168.0.29:8080/proxystat/j\_spring\_security\_check  
+ attackGET http://192.168.0.29:8080/proxystat/create.customer

- + attackGET http://192.168.0.29:8080/proxystat/showFirstsRatings.rate
- + attackGET http://192.168.0.29:8080/proxystat/.
- + attackGET http://192.168.0.29:8080/proxystat/
- + attackGET http://192.168.0.29:8080/proxystat/returnViewContact.contact

[+] Launching module permanentxss

- + http://192.168.0.29:8080/proxystat/showAllRatings.rate
- + http://192.168.0.29:8080/proxystat/postRating.rate
- + http://192.168.0.29:8080/proxystat/sendMail.contact
- + http://192.168.0.29:8080/proxystat/creation.customer
- + http://192.168.0.29:8080/proxystat/home.security?locale=en
- + http://192.168.0.29:8080/proxystat/home.security?locale=fr
- + http://192.168.0.29:8080/proxystat/j\_spring\_security\_check
- + http://192.168.0.29:8080/proxystat/create.customer
- + http://192.168.0.29:8080/proxystat/showFirstsRatings.rate
- + http://192.168.0.29:8080/proxystat/.
- + http://192.168.0.29:8080/proxystat/
- + http://192.168.0.29:8080/proxystat/returnViewContact.contact

## Report

-----

A report has been generated in the file output  
Open output/index.html with a browser to see this report