Projet Vulnérabilités

## Fonctionnalités développées

* Technologies utilisé : MySQL, Tomcat, Eclipse JEE web project, java 8, servlets, jsp/core
* Pages login, register, logout qui redirige sur login
* 2 roles possible : user, admin
* La sécurisation est rudimentaire (car login/mdp envoyé avec http/POST, en clair..):
  + mots de passe 8 car mini cryptés en BD - hashage avec classe Bcrypt opensource

(ref : <https://howtodoinjava.com/java/java-security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/> , + explications en commentaires dans le code Bcrypt.java fourni.)

* + Filtres selon un attribut de session sur toutes les pages user/\*\* et admin/\*\* selon le role
* Seule la consultation html des vulnérabilités et solutions associées est développée à ce stade
* Les logiciels, vulbérabilités et solutions seront toujours triées par ordre de dernière création
* Un utilisateur non enregistré peut créer un compte user, login/mdp entré 2x (register)
* Un utilisateur de role user peut :
  + Définir ses logiciels préférés dans ses préférences
  + Afficher la liste des vulnérabilités et solutions associées à ses logiciels préférés (logiciels affectés)
  + Changer son mot de passe (avec confirmation du mdp)
  + Se déconnecter et ainsi revenir à la page de login\*
* Un utilisateur de role admin peut :
  + Faire tout ce que fait un user
  + Consulter la liste des user et admin
  + Attribuer le role admin à un user (celui-ci doit se reconnecter)
  + Reset du password d’un user
  + Supprimer un user
  + Créer, modifier ou supprimer les vulnérabilités depuis leur liste
  + Créer, modifier ou supprimer les solutions depuis leur liste
  + Créer ou supprimer des logiciels depuis leur liste
  + Associer ou dissocier des logiciels depuis une vulnérabilité
  + Associer ou dissocier des logiciels depuis une solution
  + Associer ou dissocier des solutions depuis une vulnérabilité

## Fonctionnalités non développées

* Autres formats que HTML
* Mailing
* Tris plus avancés, par dates, par niveau d’urgence
* Amélioration visuelle des pages (le style reste rudimentaire)
* Associer ou dissocier des vulnérabilités depuis une solution

## Repository Git:

<https://github.com/gsalandrecimpa/JSPvulnerabilites>

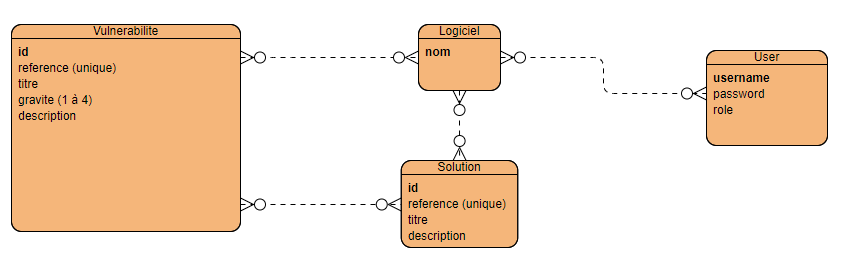
## Chemin d’accès au serveur tomcat lancé par eclipse :

<http://localhost:8080/JSPvulnerabilites/>

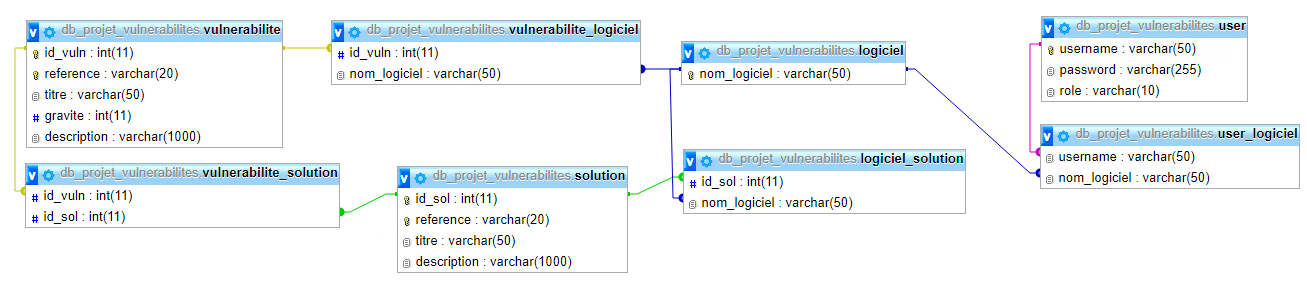
Compte admin créé par défaut par l’application : root / depart0

## Base de données

Diagramme général (capture depuis visual paradigm) :



Concepteur MySQL :



On trouve dans le répertoire projet le fichier db\_projet\_vulnerabilite.sql , export de la base initiale vide à laquelle ont été ajoutée les clauses FOREIGN KEY .. ON UPDATE RESTRICT ON DELETE CASCADE pour l’intégrité des tables de lien. Pour que ces contraintes fonctionnent, ENGINE = INNODB est positionné en créant les tables.

Un compte admin « root » (mdp « depart0 ») est créé par le modèle si non présent.

## Structure web

/\* 🡪 /login

/login 🡪 /register ; /user/vulnerabilites ou admin/vulnerabilites

/register 🡪 /user/vulnerabilites

Bandeau user 🡪 /user/vulnerabilites ; /user/logiciels ; /user/chgpw ou /logout

Bandeau admin 🡪 /admin/vulnerabilites ; /admin/logiciels ; /admin/solutions ; admin/users

/user/vulnerabilites  🡨🡪 /user/vuln

/admin/vulnerabilites 🡨🡪 /admin/editVuln

/admin/solutions 🡨🡪 /admin/editSol

Toutes les vues \*.jsp associées sont rendues non accessibles aux utilisateurs (dans /WEB\_INF/views).

Le modèle MVC est strictement respecté :

* Modèle : package ‘model’ (accès base et traitements données)
* Vues : /WEB\_INF/views/\*.jsp + /css/style.css
* Contrôleur : package ‘servlets’, une servlet par page .jsp (xx.java contrôle xx.jsp)

+ Filtres (package filters) sur les acces à /user/\* et /admin/\* selon connection et role, configurés dans web.xml

La page par défaut est /login si non connecté, /user/vulnerabilites si connecté, /admin/vulnerabilites si connecté avec role admin.

## Tests unitaire

### Package junit.model dans test

JUnit, basé sur le chainage des fonctions save\*, load\*, delete\*, link\*, unlink\*, exists\* de la classe ModelVuln. Liste des méthodes testées par groupe:

*réalisé:*

* saveUser, authenticateUser , updatePassword, resetPassword, verifyPassword, loadUsers, changeRole , existsUser, deleteUser
* saveLogiciel, deleteLogiciel, existsLogiciel, loadLogiciels
* saveSol , loadSol, returnSolFromRef, loadAllSolutions, existsSolution, deleteSol

*non réalisé:*

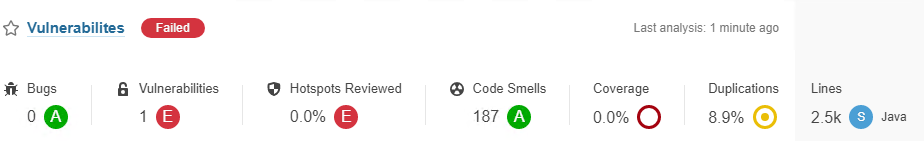
* saveVuln, loadVuln, loadAllVulnerabilites, existsVulnerabilite, deleteVuln
* updateUserLogiciels, existsUserLogiciel
* linkSolLog, saveSolLog, existsSolLog, unlinkSolLog, loadSolLogiciels, loadSolSelectLogiciels, SaveSol, loadSol
* linkVulnLog, save VulnLog, existsVulnLog, unlinkVulnLog, loadVulnLogiciels, loadVulnSelectLogiciels, loadVuln
* linkVulnSol, saveVulnSol, existsVulnSol, unlinkVulnSol, loadVulnSolutions, loadVulnSelectSolReferences, loadVuln
* existsOtherSolRef
* existsOtherVulnRef
* loadVulnerabilitesLinkedToUser

Améliorations possibles (non effectuées) :

* Pour le modèle  données d’entrée marginales (seuls des scenarios génériques sont testés à ce stade)
* Base de données de test avec données préparées et attendus, comparaison des dump de bases (bouchon sur le connecteur vers cette base spécifique).
* Les servlets restent également à tester (possible avec des bouchons vers l’API servlets notamment request/session.getAttribute, sendRedirect, Forward.. et un assert sur l’état des attributs et l’état des bouchons de controle sendRedirect et Forward ? Voir aussi pour compiler tout ça dans ant.. pas gagné)
* Pour les .jsp, on pourrait envisager une servlet générique de préparation des données basique (à base de setAttribute(« yy », new xx) ) et un Assert sur le corp de la réponse.. mais là pour le moment ça dépasse largement mes compétence et nécessiterait plus de temps et une analyse de faisabilité.

## Analyse statique

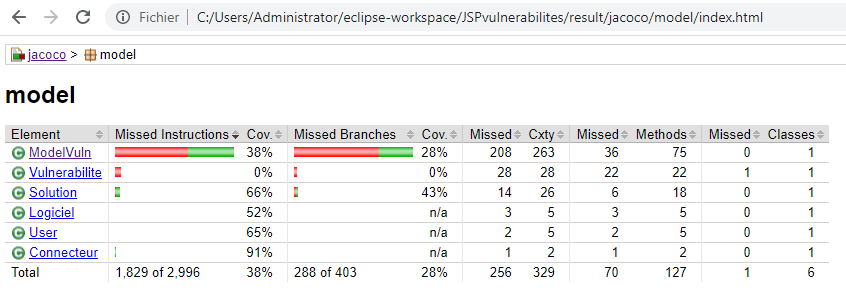
Scan et report avec SonarQube community : tous les bugs sont corrigés. La vulnérabilité concerne la DB qui est sans mot de passe pour le moment. Failed néanmoins à cause de la couverture qui n’est pas intégrée.



## Couverture

Génération d’un rapport html de couverture avec ant / jacoco :report dans <projectdir>\result\jacoco, exemple ci-dessous. (voir build.xml pour ant penser à intégrer la librairie pour le connecteur DB en plus de junit4)

Un rapport jacoco.xml est également généré, en théorie intégrable au rapport SonarQube, mais autant le sonarScanner prend bien en compte le rapport jacoco.xml, autant je ne suis pas parvenu à obtenir le résultat au niveau du serveur SonarQube. A ce stade il faut donc aller manuellement consulter <projectdir>\result\jacoco avec un navigateur internet :



## Chaine d’intégration

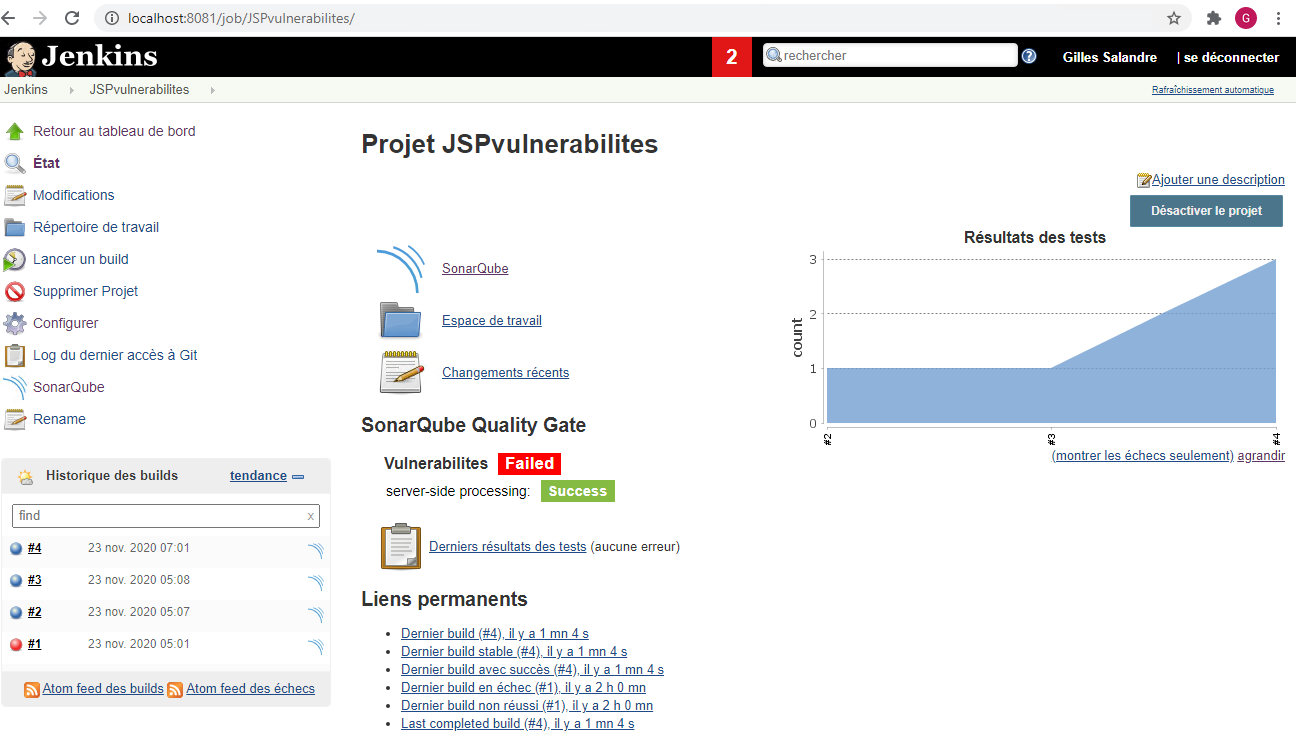
GitHub / ant / SonarQube / Jenkins

(ne couvre que le package model pour le moment.)

Github : https://github.com/gsalandrecimpa/JSPvulnerabilites

Lancer Jenkins sur le port 8081 car tomcat utilise 8080. (fichier jenkins.xml à changer)

Résultat :



## Tests fonctionnels

Non réalisés à part le scenario ci-dessous.

Intégration d’un seul scenario Selenium vers Junit pour l’exemple : package junit.selenium dans test.

Quelques corrections sont nécessaire dans le code généré par selenium (typo dans chaines de caractères, ajouts intempestifs de \..) + ajout de clear() entre les click() et sendKeys(), mais au final ça s’execute bien.

* Scenario1 – création/administration de compte/acces:
  + Prérequis : destruction au préalable du compte testselenium (appel à model.deleteUser)
  + Page login, register, « root »/ « monpassword » -> err « le compte existe déjà »
  + « testselenium »/ « monpassword » -> accès à la page user/vulnerabilites
  + Tentative de charger admin/ vulnerabilitesdans la barre d’adresse -> echec accès , on revient sur la page d’accueil user/vulnerabilites 🡪 pas réussi avec selenium
  + Logout -> login « root » / « depart0 » -> admin/users -> setadmin testselenium -> initpwd

testselenium (xpath=//td[text()='testselenium']//following::td[2]//child::button[@name='resetpwd'])

-> logout

* + Login “testselenium”/ « monpassword » -> echec (err)
  + Login “testselenium”/“depart0“ -> succes -> acces à admin/vulnerabilites

\*\*fin\*\*

