

SonarQube



Bureau E204


Plan du cours

- Introduction
- Tests dynamiques et Tests statiques
- SonarQube
 - Définition
 - Caractéristiques
 - Fonctionnalités
 - Rapports
 - Architecture
 - Installation (Plugin / Image Docker)
 - Utilisation (Intégration avec l'outil Jenkins)
 - Analyse des résultats
 - Exemple
- Travail à faire

Introduction

Une application fonctionnelle peut comporter :

- ✓ Un code dupliqué à plusieurs endroits.
- ✓ Les méthodes ne sont pas commentées.
- ✓ Une négligence des bonnes pratiques de développement:

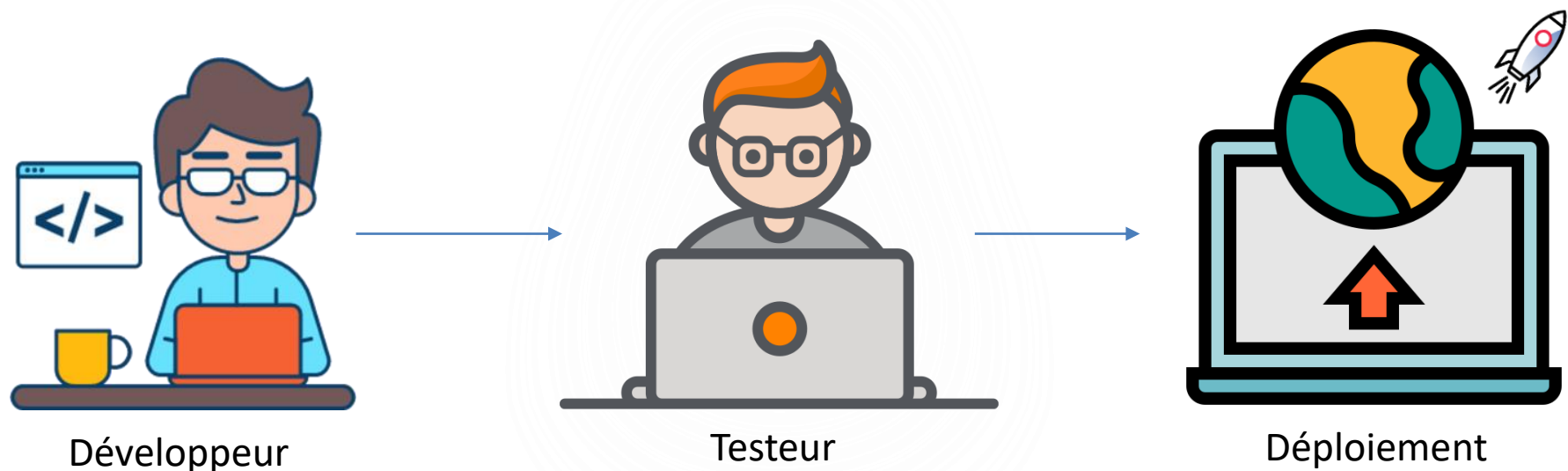
(CamelCase  `newString;` , `System.out.println("Hello World");` , etc ...)

Des problèmes peuvent provenir :

- ✓ Des bugs (Exemple : Appliquer un getter sur un objet nul)
- ✓ Des tests unitaires mal développés
- ✓ Classes trop volumineuses à découper

Tests dynamiques et Tests statiques

- Les **tests** font partie de cycle de vie du développement d'une application donnée.
- Les tests visent à s'assurer que le code qui sera déployé est de **bonne qualité, sécurisé** et ne présente pas de **bugs** (environ **30%** du temps de développement doit être consacré aux tests).

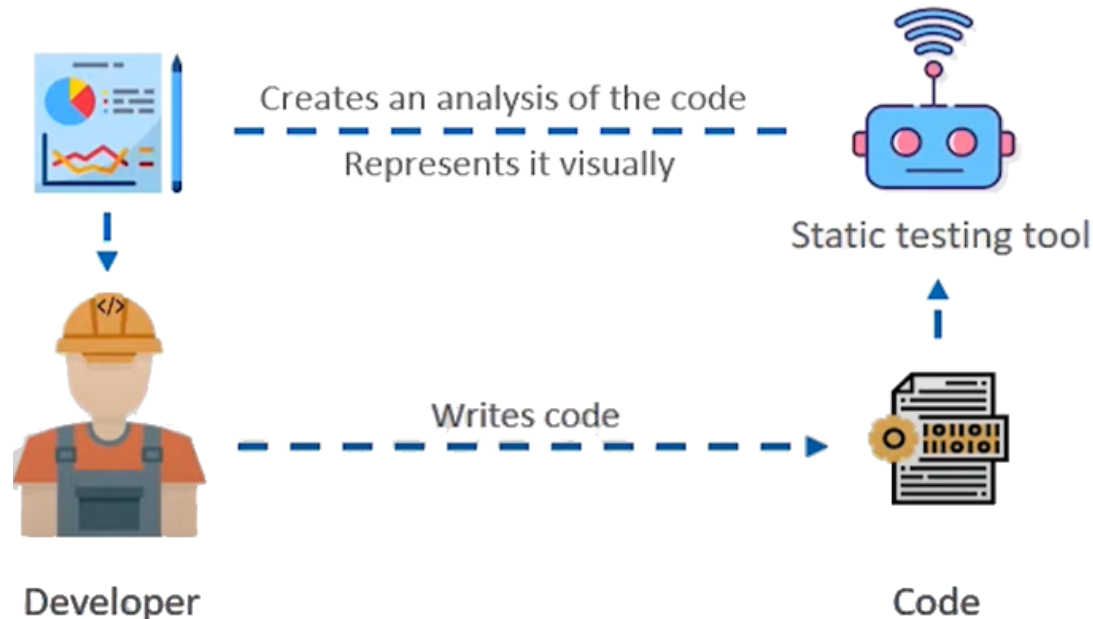


Tests dynamiques et Tests statiques

- Les **tests dynamiques** : Ces tests sont faits alors que l'application tourne, pour détecter les **dysfonctionnements** (fonctionnalités mal implémentées, DB inaccessible, ...) : Tests Unitaires (JUnit par exemple).
- Les **tests statiques** : Ces tests sont faits sur le code source, avant de l'exécuter. Il s'agit d'analyser le code pour détecter les écarts aux **bonnes pratiques de développement** (absence de logs, absence de commentaires.): SONARQUBE fait ce type de tests.

SonarQube - Définition

- **SonarQube** est un outil de test statique, open source, utilisé pour analyser la qualité du code source, selon des règles prédéfinies. Il permet donc l'inspection en continue de la qualité de votre code (Code Review automatique).
- **SonarQube** s'appuie sur l'analyse statique du code.



SonarQube - Caractéristiques

- **SonarQube** peut être utilisé avec une vingtaine de langages (Java, .Net (C#), Python, PHP, Cobol, JavaScript, ...)
- Il permet de détecter **les défauts de codage** (code jamais utilisé, dupliqué, sans commentaires, sans tests unitaires, sans gestion d'exception, non sécurisé,).
- Il nous permet de choisir **les règles à activer** lors de l'analyse de notre code.



SonarQube - Fonctionnalités

Appart la qualité de code, **SonarQube** s'intéresse à plusieurs aspects tels que :

- ✓ La détection rapide des erreurs
- ✓ Les applications durables
- ✓ Une meilleure productivité
- ✓ La flexibilité
- ✓ La réduction des coûts
- ✓ L' évaluation de la couverture du code par les tests unitaires
- ✓ Des analyses entièrement automatisées (intégration avec Jenkins et Maven)

SonarQube - Rapports

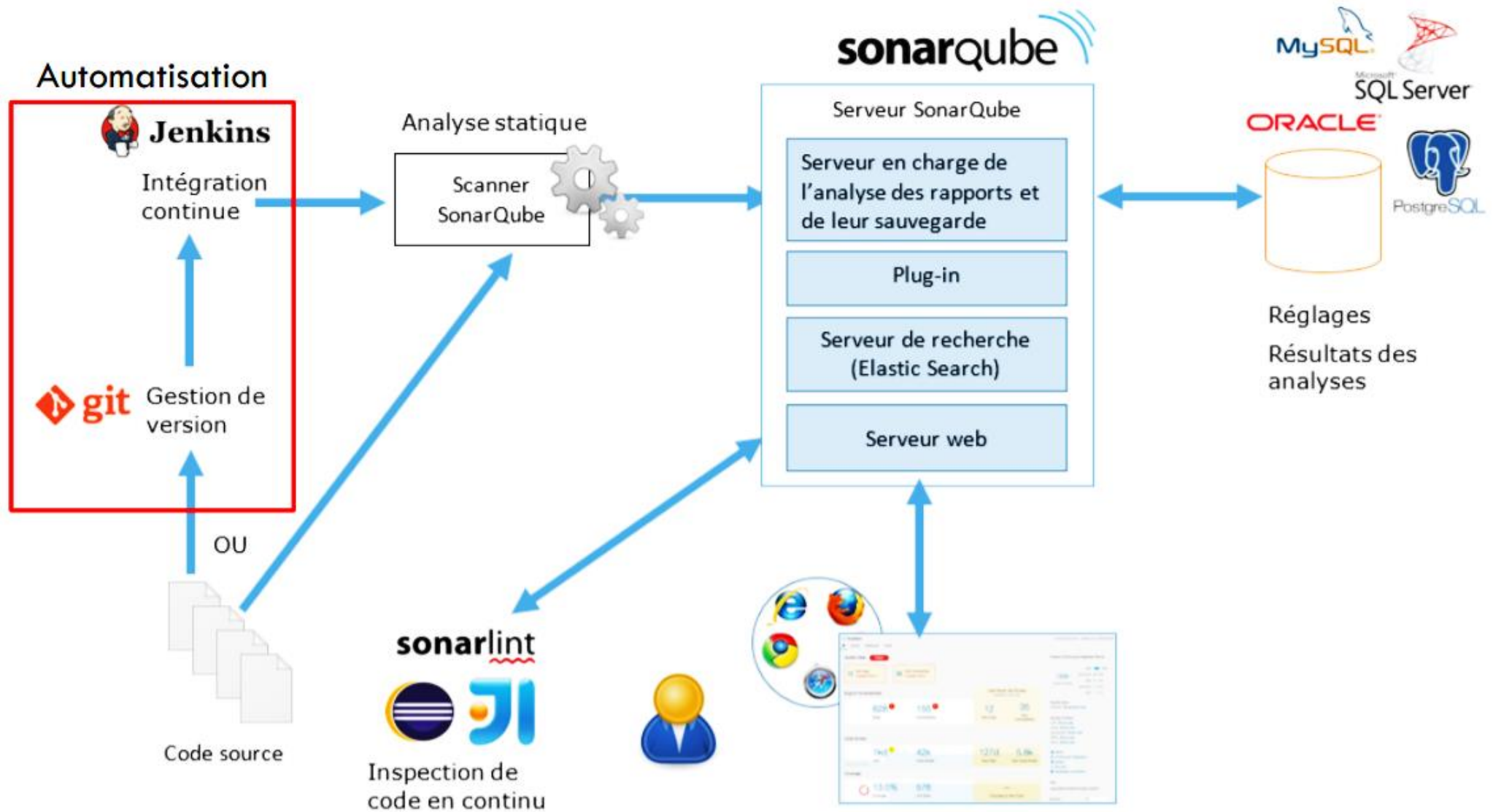
SonarQube génère différents types de rapports qui s'intéressent à différents aspects de qualité :

- Architecture de l'application
- Test unitaire
- Codes dupliqués
- Bugs potentiels
- Code complexe
- Règles de programmation
- Commentaires

SonarQube - Couverture du code

- **SonarQube** permet de connaître le nombre de tests unitaires joués, le taux de succès et la couverture de ces tests.
- La couverture de tests correspond au pourcentage de lignes de code du projet utilisés pour l'analyse.
- Une couverture basse est l'indice d'une carence dans les tests et engendre des interrogations sur la fiabilité du produit.
- Un taux de succès des tests qui n'est pas à 100% est un indice que un ou plusieurs tests unitaires (ou fonctionnalités) ont été mal développés.

SonarQube - Architecture



SonarQube - Architecture

- SonarQube est constituée de plusieurs composants :
 - **Un exécuteur** qui lance les outils d'analyse de code sources externes et internes ;
 - **Des plugins** qui étendent SonarQube : support d'autres langages (PHP), métriques supplémentaires (couplage avec JDepend), utilisation d'analyseurs externes (PMD, Findbugs, ...) ;
 - **Un serveur Sonar** qui :
 - o Agrège les résultats des analyses et les enregistre dans la base ;
 - o Permet aux développeurs de consulter les résultats des analyses des projets depuis un serveur web (tableaux de bords) ;
 - o Gère les recherches faites depuis l'interface web (ElasticSearch)

SonarQube - Architecture

- **Une base de données** (par défaut H2, mais il peut utiliser MySQL, PostgreSQL, SQLServer, Oracle) pour stocker :
 - o Les réglages de configuration ;
 - o L'historique des analyses des projets surveillés par sonar ;
- Il est possible d'automatiser l'analyse avec un serveur d'intégration continue (Exemple: Jenkins, Travis).
- SonarQube permet l'inspection de code en continu (intégration dans les IDE, vérification de la qualité depuis la dernière version ou la dernière analyse, notifications par email), ce qui permet de détecter les problèmes dès leur introduction dans le code, avant que le coût de remédiation soit élevé.

SonarQube - Installation

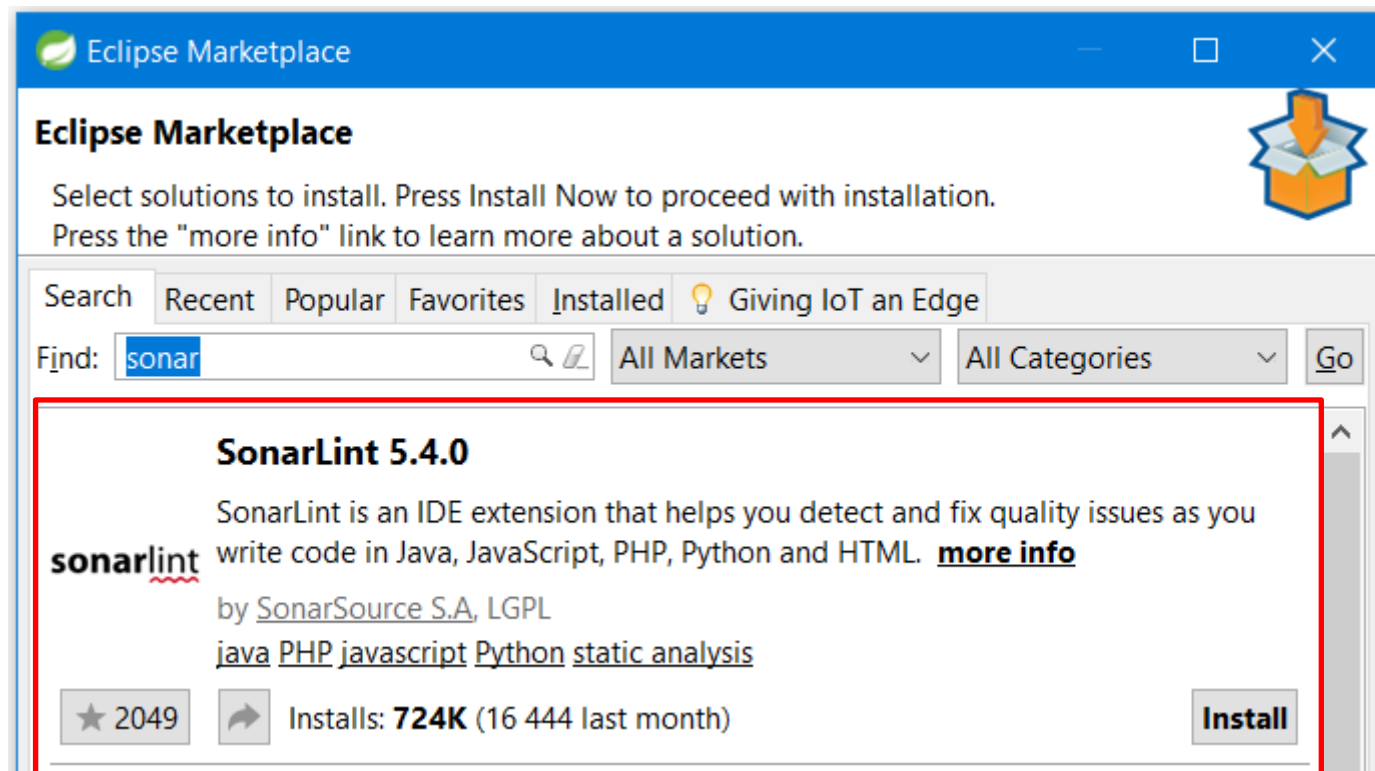
SonarQube est accessible de divers manières :

- Plugin au niveau de l'IDE (STS, IntelliJ, ...)
- Image Docker
- ...



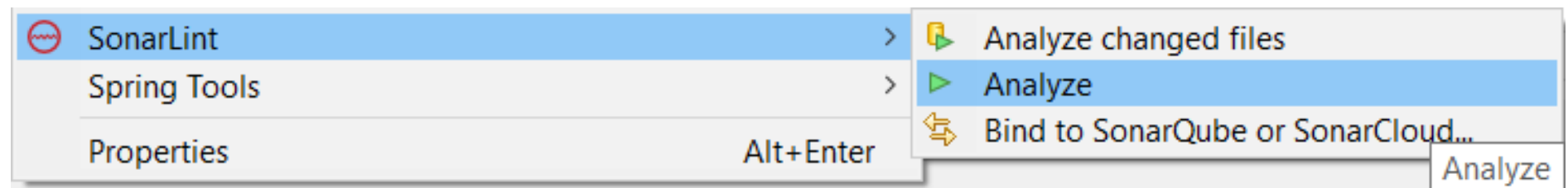
SonarQube – Installation (Plugin)

- Dans STS, aller dans Help -> Eclipse Marketplace, chercher « sonar », installer le plugin « SonarLint », accepter la licence, accepter de redémarrer STS



SonarQube – Installation (Plugin)

- Pour faire l'analyse, Ouvrir un de vos projets sur STS, bouton droit et choisir « Sonar Lint » -> Analyze



- Vous aurez les différentes suggestions de correction/amélioration de code:

A screenshot of the SonarLint Report panel in an IDE. The panel shows a list of 6 items with columns for Resource and Description. The items are listed as follows:

Resource	Description
Calcul.java	Complete the task associated to this TODO comment.
Calcul.java	Replace this use of System.out or System.err by a logger.
Calcul.java	Remove this unused "j" local variable.
Calcul.java	Remove this useless assignment to local variable "j".
CalculTest.java	Remove this assertion from production code.
CalculTest.java	This block of commented-out lines of code should be removed.

SonarQube – Installation (Image Docker)

- Récupérez l'image Docker «sonarqube:**8.9.7-community**» de Dockerhub

```
docker pull sonarqube:8.9.7-community
```

Attention: La version 8.9.7-community et non la latest.

```
[vagrant@localhost ~]$ docker pull sonarqube:8.9.7-community
```

- Lancez Sonarqube :

```
docker run -d -p 9000:9000 sonarqube:8.9.7-community
```

```
2022.09.27 20:14:36 INFO app[][o.s.a.SchedulerImpl] Process[ce] is up
2022.09.27 20:14:36 INFO app[][o.s.a.SchedulerImpl] SonarQube is up
```

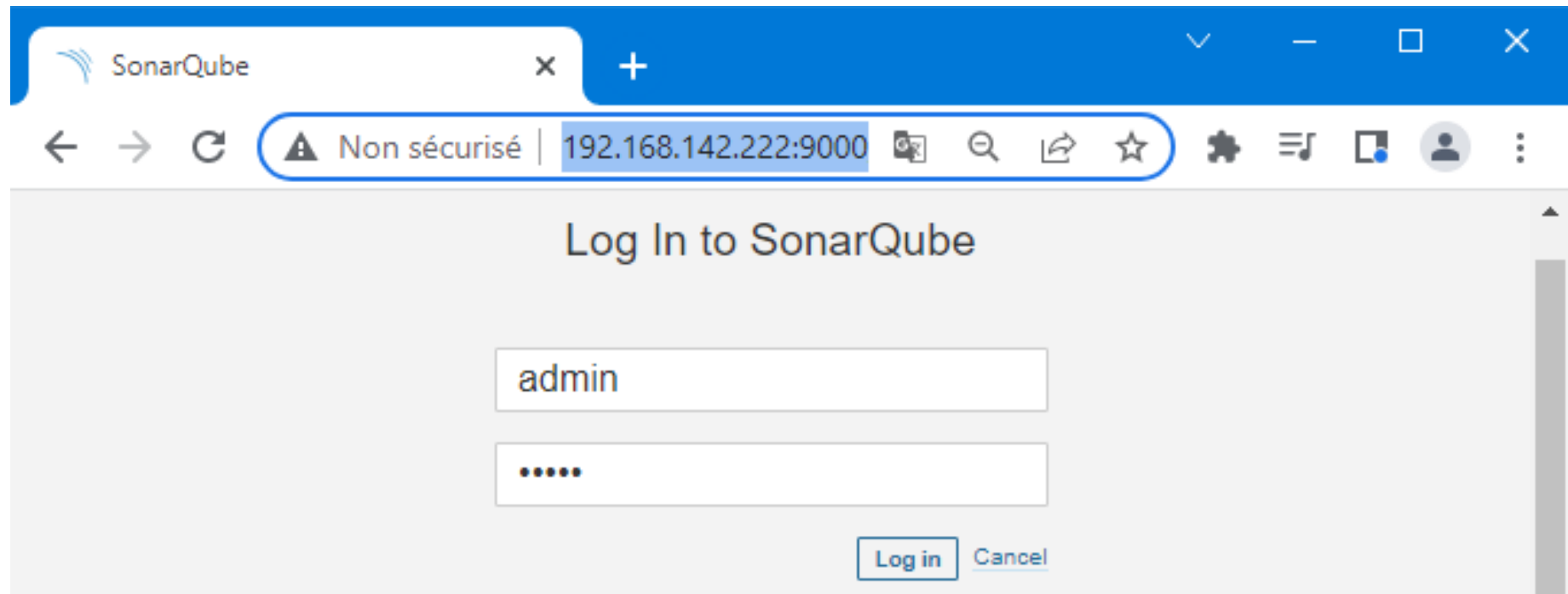
Optionnel: C'est possible de lancer Sonar en background (option -d ou un docker start sur le conteneur).

SonarQube – Utilisation (Image Docker)

- Sur votre machine Windows, aller à l'url

<http://<@ip>:9000>

et se connecter avec « admin / admin » :



- Changer le mot de passe à sonar par exemple (admin / sonar).

SonarQube – Utilisation (Image Docker)

- **Attention:** Si vous arrêtez le conteneur sonarqube (CTRL S ou en arrêtant la VM), il ne faut pas lancer un docker run sur l'image sonarqube, car cela créera un nouveau conteneur. Il faut juste faire :

```
> Sélection vagrant@localhost:~  
[vagrant@localhost ~]$ sudo chmod 666 /var/run/docker.sock  
[vagrant@localhost ~]$ docker ps -a  
CONTAINER ID        IMAGE                                     COMMAND  
ba73cc1e77eb        sonarqube:8.9.7-community              "bin/run.s  
8811511288         8-5858121685
```

- Puis faire un docker start id-conteneur-sonarqube et attendre quelques minutes le temps que tout se lance (Elasticsearch, Sonarqube, ...) :

```
[vagrant@localhost ~]$ docker start ba73cc1e77e  
ba73cc1e77eb  
[vagrant@localhost ~]$
```

SonarQube – Utilisation (Avec Jenkins)

- Puisque nous n'avons pas de projets à analyser sur notre VM, nous allons utiliser Jenkins, pour récupérer un projet de Git, puis l'analyser avec Sonarqube :
- Se connecter à **Jenkins** via l'url <http://<<@ip>:8080>
- Sur le **pipeline** Jenkins déjà créé, récupérer le code de votre projet de Github en ajoutant le stage suivant dans le script Groovy (mettez l'URL de votre projet ou testez avec le repo Git ci-dessous) :

```
stage ('GIT') {  
    steps {  
        echo "Getting Project from Git";  
        .....  
    }  
}
```

SonarQube – Utilisation (Avec Jenkins)

- Sur le même pipeline, lancer les commandes Maven **clean** et **compile** pour compiler le code de votre projet récupéré de Git (sh « ... ») :

```
stage('MVN CLEAN') {  
    steps {  
        sh '.....'  
    }  
}
```

```
stage('MVN COMPILE') {  
    steps {  
        .....  
    }  
}
```

SonarQube – Utilisation (Avec Jenkins)

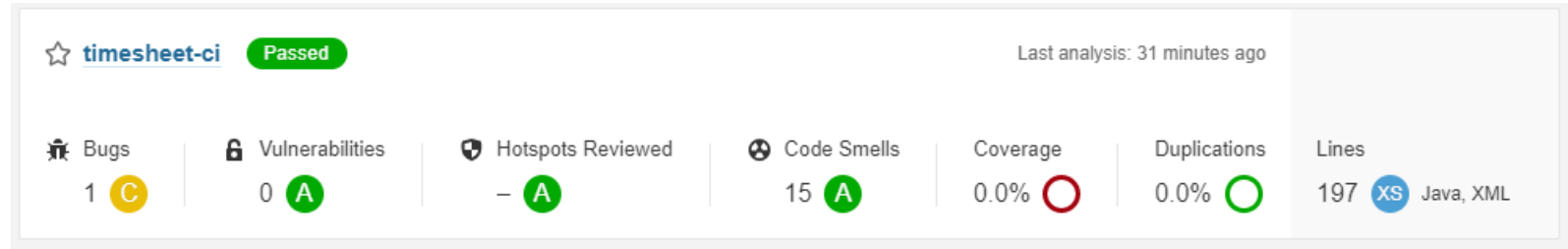
- Sur le même pipeline, lancer la commandes Maven d'analyse de code mvn **sonar:sonar** pour analyser la qualité de votre code et envoyer le rapport au serveur Sonarqube (Mettez le mot de passe de votre Sonarqube) :

```
stage('MVN SONARQUBE') {  
    steps {  
        .....  
    }  
}
```

- Lancer le Job via Jenkins :

SonarQube - Analyse des résultats

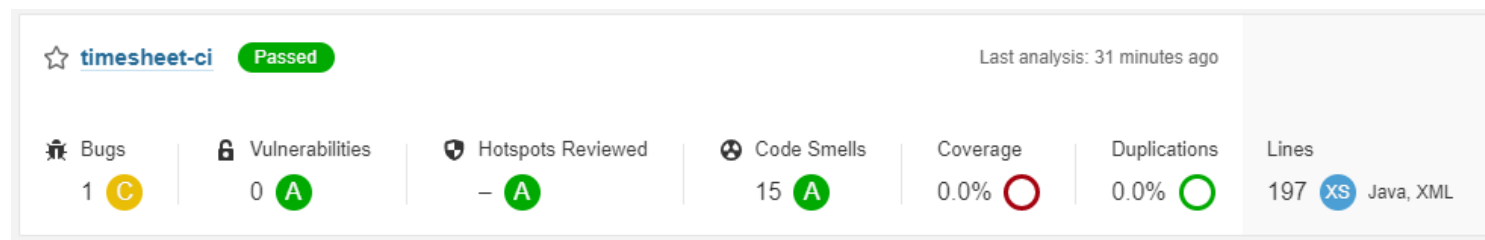
- Aller dans <http://<ip-vm>:9000> et regarder le résultat de l'analyse de votre projet :



- Sur l'interface ci-dessus, il suffit de cliquer sur le projet « timesheet » pour accéder aux détails de toute cette analyse :
- Sonarqube détecte automatiquement le **langage (Java + XML)** dans notre cas)
- Sonarqube vérifie si le développeur a mis du **code dupliqué** dans plusieurs endroit (source d'erreur) : 0% dans notre cas

SonarQube - Analyse des résultats

- Aller dans <http://<@ip>:9000> et regarder le résultat de l'analyse de votre projet :



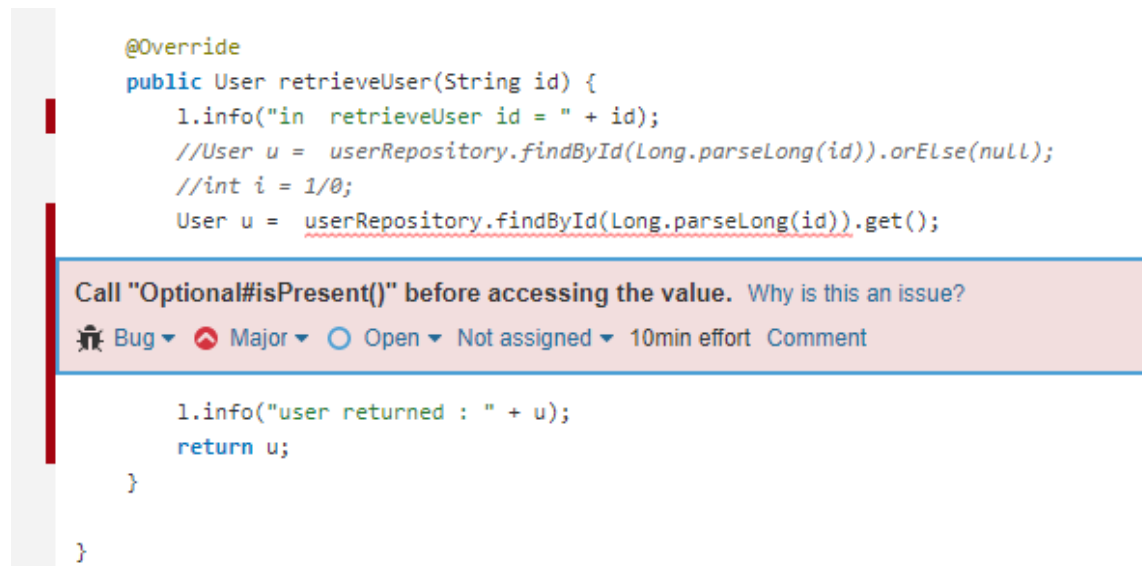
- Métriques par défaut:

Score	Fiabilité	Sécurité	Maintenabilité
A	0	0	ratio dette technique $\leq 5\%$
B	≥ 1 mineur	≥ 1 mineure	$6\% \leq \text{ratio dette technique} \leq 10\%$
C	≥ 1 majeur	≥ 1 majeure	$11\% \leq \text{ratio dette technique} \leq 20\%$
D	≥ 1 critique	≥ 1 critique	$21\% \leq \text{ratio dette technique} \leq 50\%$
E	≥ 1 bloquant	≥ 1 bloquante	ratio dette technique $\geq 51\%$
Issue	Bug	Vulnerability	Code smell

- Pour modifier les métriques, il suffit juste accéder à l'onglet « Quality gate » et créer un nouveau portail de qualité

SonarQube - Analyse des résultats

- **Bug** : Un code qui peut engendrer un mauvais comportement de l'application (un null pointer exception par exemple), dans notre cas :



- **Vulnerability** : Faille de sécurité dans notre code.
- **Hotspots Reviewed** : Code à revoir pour être sûr que ce n'est pas un faille de sécurité.

SonarQube - Analyse des résultats

- Sonarqube affiche si le code a été exécuté par les outils de tests (comme JUnit). Sonarqube ne fait pas l'analyse lui-même mais se base sur d'autres outils comme JaCoCo (c'est pour cela qu'on a 0% de **Coverage** puisque JaCoCo n'est pas ajouté à notre projet).
- **Code Smells** : Ce n'est pas un bug, mais c'est un code qui peut retarder l'équipe de développement ou l'équipe de support quand ils essaient de comprendre ou modifier le code (exemple : beaucoup de commentaires ou des imports non utilisés) :



SonarQube – Exemple (Erreur/Correction)

CentreCommercialService.java Boutique.java ✕

```
1 package tn.esprit.spring.entity;
2
3 import java.io.Serializable;
4 import java.util.ArrayList;
5 import java.util.HashSet;
6 import java.util.List;
7 import java.util.Set;
8
9 import javax.persistence.*;
10
11 import com.fasterxml.jackson.annotation.JsonIgnore;
12
13 @Entity
14 public class Boutique implements Serializable {
15
```

Problems Javadoc Declaration Console SonarLint Report ✕

36 items















Resource	Date	Description
ApplicationRestController.java		⚠️📌 Make client a static final constant or non-public and provide accessors if needed.
ApplicationRestController.java		⚠️📌 Make idBoutiques a static final constant or non-public and provide accessors if needed.
ApplicationRestController.java		🔒📌 Replace this persistent entity with a simple POJO or DTO object.
Boutique.java		⚠️📌 Remove this unused import 'java.util.HashSet'.
Boutique.java		⚠️📌 Remove this unused import 'java.util.Set'.

SonarQube – Exemple (Erreur/Correction)

```
55 @Transactional
56 public void assignFournisseurToProduit(Long fournisseurId, Long produitId)
57     Fournisseur fournisseur = fournisseurRepository.findById(fournisseurId)
58     Produit produit = produitRepository.findById(produitId).orElse(null);
59     produit.getFournisseurs().add(fournisseur);
60 }
61
62 }
```

Console Tasks Debug Servers (x)= Variables Breakpoints Outline JUnit Git Repositories Git Staging SonarLint Report

items

resource	Date	Description
FactureRestController.java		  This block of commented-out lines of code should be removed.
FactureRestController.java		  This block of commented-out lines of code should be removed.
FactureRestController.java		  This block of commented-out lines of code should be removed.
FactureRestController.java		  Replace this persistent entity with a simple POJO or DTO object.
FactureServiceImpl.java	4 days ago	  A "NullPointerException" could be thrown; "produit" is nullable here.
FactureServiceImpl.java	28 days ago	  A "NullPointerException" could be thrown; "facture" is nullable here.
FactureServiceImpl.java		  Remove this unnecessary cast to "List".

SonarQube – Exemple (Erreur/Correction)

```
@Override
public String addUser(User user) {
    uR.save(user);
    return "done";
}

@Override
public String updateUser(User user) {
    uR.save(user);
    return "done";
}
```

Problems @ Javadoc Declaration Console SonarLint Report

20 items

Resource	Date	Description
UserRestController.java		Remove this unused import 'org.springframework.context.annotation.ComponentScan'.
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.DELETE)" with "@DeleteMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.POST)" with "@PostMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.PUT)" with "@PutMapping" [+1 location]
UserService.java	few seconds ago	Update this method so that its implementation is not identical to "addUser" on line 20. [+1 location]
UserService.java		Complete the task associated to this TODO comment.

SonarQube – Exemple (Erreur/Correction)

```
@RequestMapping(value = "/findAll", method = 1 RequestMethod.GET, produces = { MediaType.APPLICATION_JSON_VALUE,
    MediaType.APPLICATION_JSON_VALUE })
public List<User> findAll() {
    return uS.findAll();
}
```

Problems @ Javadoc Declaration Console SonarLint Report SonarLint Issue Locations

20 items

Resource	Date	Description
UserRestController.java		Remove this unused import 'org.springframework.context.annotation.ComponentScan'.
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.DELETE)" with "@DeleteMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]

SonarQube – Exemple (Erreur/Correction)

```
package com.example.demo;

import org.junit.Test;

@RunWith(SpringRunner.class)
@SpringBootTest
public class SpringBootTestApplicationTests {

    @Test
    public void contextLoads() {

    }

}
```

Problems @ Javadoc Declaration Console SonarLint Report SonarLint Issue Locations		
20 items		
Resource	Date	Description
SpringBootTestApplication		❗ Add at least one assertion to this test case.
UserControl.java		❗ Complete the task associated to this TODO comment.
UserControl.java		❗ Complete the task associated to this TODO comment.

SonarQube – Exemple (Erreur/Correction)

```
@Override
public User doLogin(String email, String passwd) {
    // TODO Auto-generated method stub
    return uR.findByEmailAndPasswd(email, passwd);
}
```

Problems @ Javadoc Declaration Console SonarLint Report SonarLint Issue Locations		
20 items		
Resource	Date	Description
UserRestController.java		Remove this unused import 'org.springframework.context.annotation.ComponentScan'.
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.DELETE)" with "@DeleteMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.POST)" with "@PostMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.PUT)" with "@PutMapping" [+1 location]
UserService.java	10 minutes ago	Update this method so that its implementation is not identical to "addUser" on line 20. [+1 location]
UserService.java		Complete the task associated to this TODO comment.

SonarQube – Exemple (Erreur/Correction)

```
// http://localhost:8081/spring-data-ipa/doLogin
@RequestMapping(value = "/doLogin", method = RequestMethod.GET, produces = { MediaType.APPLICATION_JSON_VALUE,
    MediaType.APPLICATION_JSON_VALUE })
public User doLogin(@RequestParam("email") String email, @RequestParam("passwd") String passwd) {
    return uS.doLogin(email, passwd);
}

// http://localhost:8081/spring-data-ipa/deleteUser/{id}
@RequestMapping(value = "/deleteUser/{id}", method = RequestMethod.DELETE, produces = {
    MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_JSON_VALUE })
public String deleteUser(@PathVariable("id") long id) {
    return uS.deleteUser(id);
}
```

Problems	@ Javadoc	Declaration	Console	SonarLint Report	SonarLint Issue Locations
23 items					
Resource	Date	Description			
UserRestController.java	few seconds ago	⊗⬆ This block of commented-out lines of code should be removed.			
UserRestController.java	few seconds ago	⊗⬆ This block of commented-out lines of code should be removed.			
UserRestController.java		⊗⬆ Remove this unused import 'org.springframework.context.annotation.ComponentScan'.			
UserRestController.java		⊗⬆ Replace "@RequestMapping(method = RequestMethod.DELETE)" with "@DeleteMapping" [+1 location]			
UserRestController.java		⊗⬆ Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]			
UserRestController.java		⊗⬆ Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]			
UserRestController.java		⊗⬆ Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]			
UserRestController.java		⊗⬆ Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]			

SonarQube – Exemple (Erreur/Correction)

```
public enum Role {  
    Administrateur, Directeur, Enseignant  
}
```

Problems	Javadoc	Declaration	Console	SonarLint Report	SonarLint Issue Locations
26 items					
Resource	Date	Description			
Role.java	few seconds ago	⚠️🔴 Rename this constant name to match the regular expression '^[A-Z][A-Z0-9]*([A-Z0-9]+)*\$'.			
Role.java	few seconds ago	⚠️🔴 Rename this constant name to match the regular expression '^[A-Z][A-Z0-9]*([A-Z0-9]+)*\$'.			
Role.java	few seconds ago	⚠️🔴 Rename this constant name to match the regular expression '^[A-Z][A-Z0-9]*([A-Z0-9]+)*\$'.			
SpringBootTestApplication		⚠️🔴 Add at least one assertion to this test case.			
UserControl.java		⚠️🔵 Complete the task associated to this TODO comment.			
UserControl.java		⚠️🔵 Complete the task associated to this TODO comment.			

SonarQube – Exemple (Erreur/Correction)

```
@Override
public String addUser(User user) {
    uR.save(user);
    System.out.println(user.toString());
    return "done";
}
```

Problems @ Javadoc Declaration Console SonarLint Report SonarLint Issue Locations		
26 items		
Resource	Date	Description
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.POST)" with "@PostMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.PUT)" with "@PutMapping" [+1 location]
UserService.java	few seconds ago	Replace this use of System.out or System.err by a logger.
UserService.java		Complete the task associated to this TODO comment.

SonarQube – Exemple (Erreur/Correction)

```
@Override
public String updateUser(User user) {
    if (uR.save(user) != null) {
        return "done";
    }
    else {

    }
    return "done";
}
```

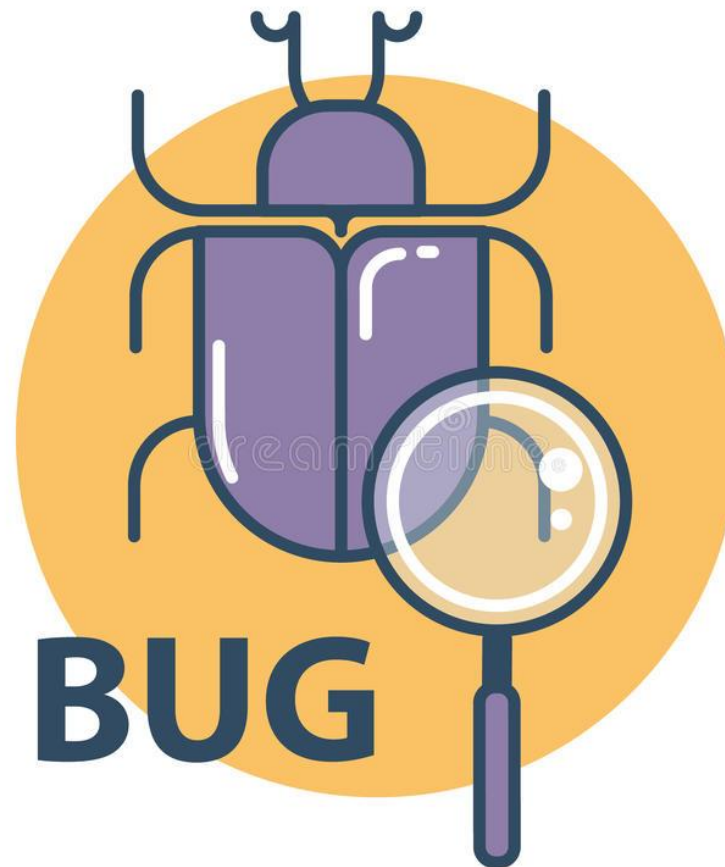
Problems @ Javadoc Declaration Console SonarLint Report SonarLint Issue Locations

28 items

Resource	Date	Description
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.GET)" with "@GetMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.POST)" with "@PostMapping" [+1 location]
UserRestController.java		Replace "@RequestMapping(method = RequestMethod.PUT)" with "@PutMapping" [+1 location]
UserService.java	few seconds ago	Change this condition so that it does not always evaluate to "true" [+2 locations]
UserService.java	few seconds ago	Either remove or fill this block of code.
UserService.java	2 minutes ago	Replace this use of System.out or System.err by a logger.
UserService.java		Complete the task associated to this TODO comment.

Travail à faire

- Installer Sonar et essayer de fixer le maximum d'erreurs détectées sur le projet.



Si vous avez des questions, n'hésitez pas à nous contacter :

Département Informatique
UP ASI

Bureau E204

SonarQube

sonarqube 