Une démarche qualité basée sur des analyses régulières Sonar (V2)

Ce document présente la démarche pour vous connecter au serveur SonarQube qui vous permettra de suivre la qualité de vos développements.

Pour l'instant on se focalise sur la partie back-end, mais toute proposition pour connecter/améliorer la qualité de votre front-end sera la bienvenue.

Configuration du serveur

Le serveur est accessible sur le sous-réseau UGA (**VPN ou tunneling uniquement**) à l'adresse :

http://im2ag-sonar.u-ga.fr:9000/

Les groupes sont déjà configurés en fonction de votre numéro de groupe noté X_Y ci-après:
Login : mgX_Y
Password : mgY

Sur le serveur, il s'agit de créer un nouveau projet et d'obtenir une clé d'accès qui nous servira pour l'accès distant. Choisir un nom de projet cohérent avec les conventions maven « <groupid> :<artifactid> ». Définir l'artifactid en suivant la convention « yourproject_gX_Y ».

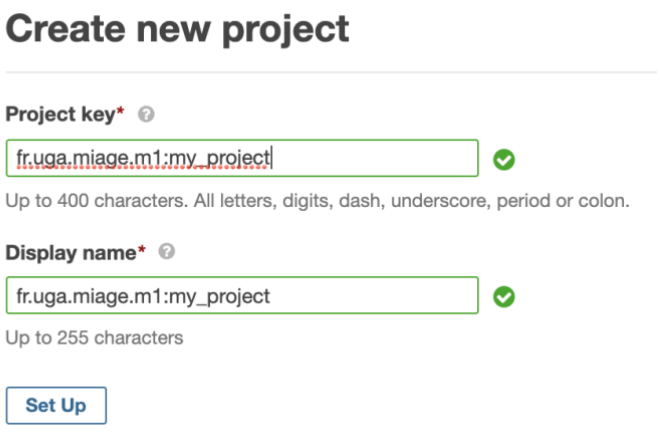

Figure 1 : Création d'un projet

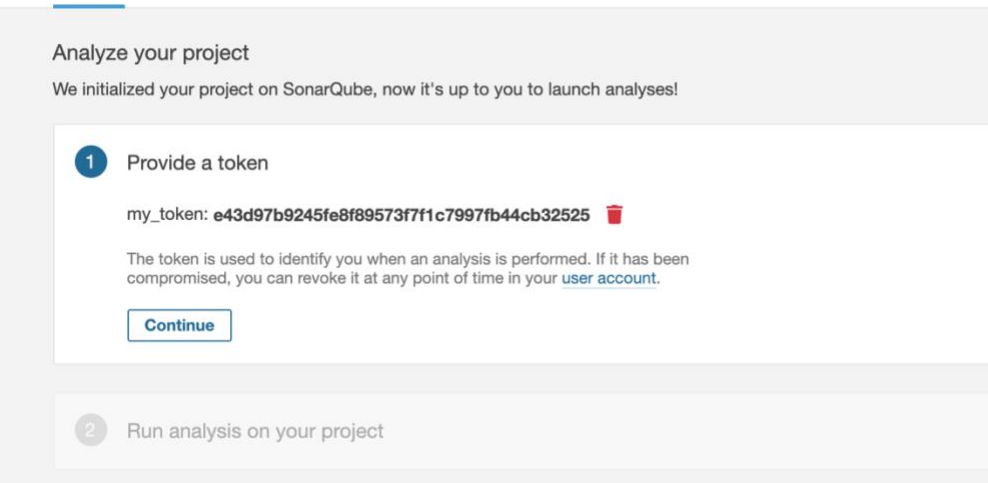

Figure 2 : Création du token

Configuration du projet Maven

Vous devez partir d'une structure de projet Maven :

Le plus simple, importer dans Eclipse le template de projet disponible sur Moodle (Import…/Maven/Existing Maven Project)
https://im2ag-moodle.univ-grenoble-alpes.fr/mod/resource/view.php?id=26599

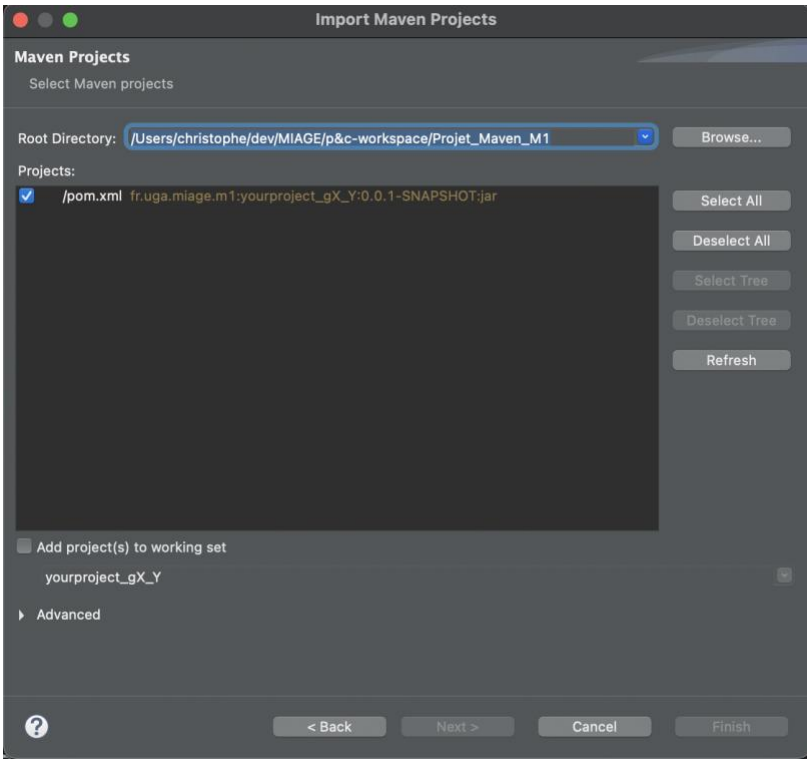

Figure 3 : Import d'un projet Maven

Sinon, créer un nouveau projet Eclipse/Maven dans lequel vous mettrez vos sources et vos tests (New/Other/Maven/Maven Project).

Sélectionnez Create a simple projet …,

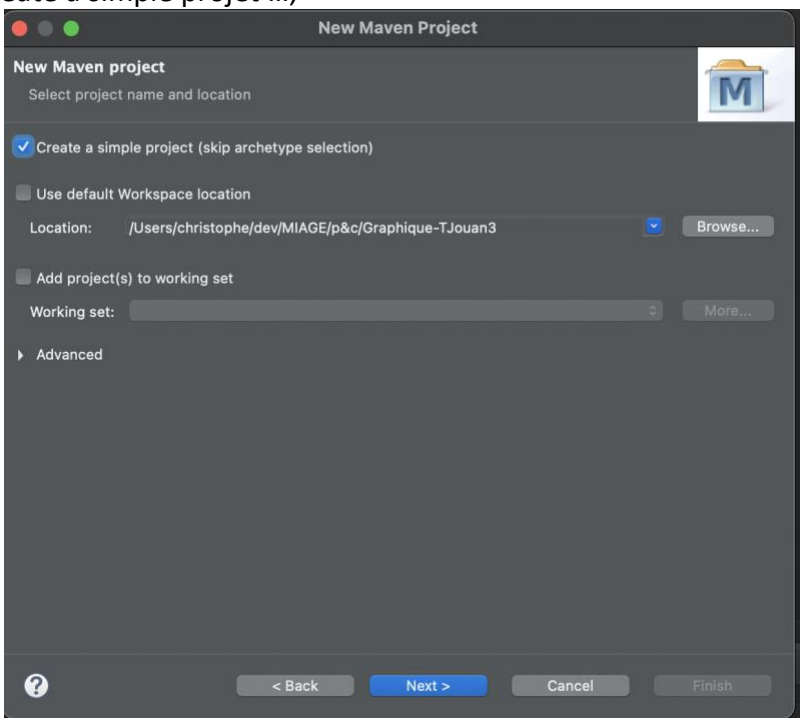

Figure 4 : Création d'un projet Maven

Ce qui vous permettra de saisir votre group_id et votre artifact_id.

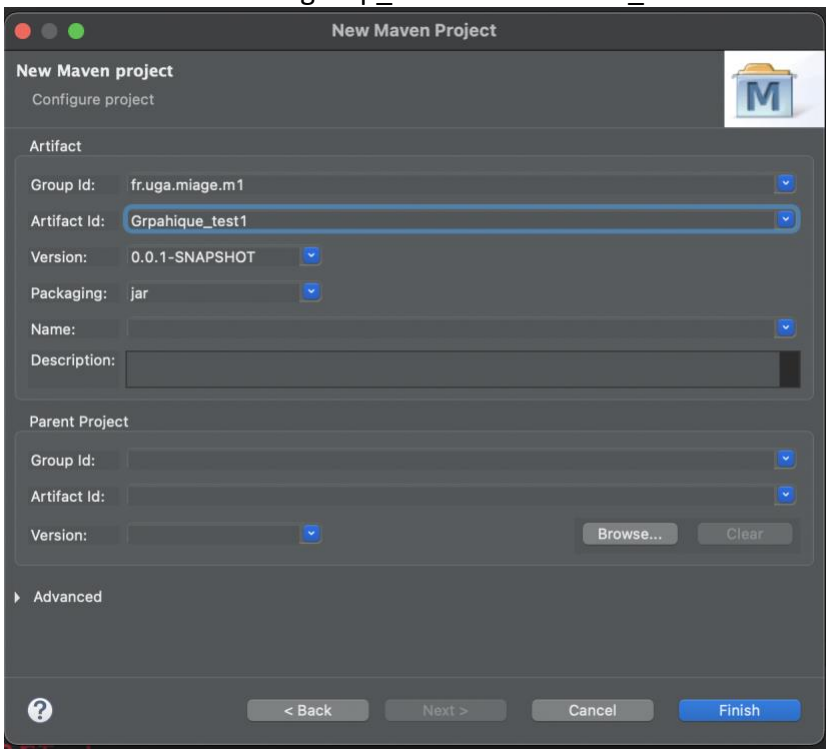

Figure 5 : Group Id et Artifact Id

Une fois le projet créé, vous pouvez remplacer le pom.xml par celui-ci-après. On ajoute aussi une dépendance vers JUnit et un plugin (surefire) pour les tests ainsi qu'un plugin pour la couverture de code (Jacoco).

Pensez à modifier l'artifactId !

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>fr.uga.miage.m1</groupId>
        <artifactId>yourproject_gX_Y</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <properties>
                <project.build.sourceEncoding>ISO-8859-1</project.build.sourceEncoding>
                <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <sonar.host.url>http://im2ag-sonar.u-ga.fr:9000</sonar.host.url>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>org.junit.jupiter</groupId>
                        <artifactId>junit-jupiter-engine</artifactId>
                        <version>5.4.0</version>
                        <scope>test</scope>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>
                                <artifactId>maven-compiler-plugin</artifactId>
                                <version>3.8.1</version>
                        </plugin>
                        <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        </plugin>
       <plugin>
```

```
            <groupId>org.jacoco</groupId>
            <artifactId>jacoco-maven-plugin</artifactId>
            <version>0.8.7</version>
            <executions>
              <execution>
                <goals>
                  <goal>prepare-agent</goal>
                </goals>
              </execution>
              <execution>
                <id>generate-code-coverage-report</id>
                <phase>test</phase>
                <goals>
                  <goal>report</goal>
                </goals>
              </execution>
            </executions>
          </plugin>
        </plugins>
            </build>

</project>
```

Attention, la structure d'un projet Maven est différente de celle d'un projet Java.

- Les sources sont sous src/main/java
- Les tests sont sour src/test/java

Pour vérifier que votre projet fonctionne, une seule commande qui compile et teste votre programme:

>mvn install

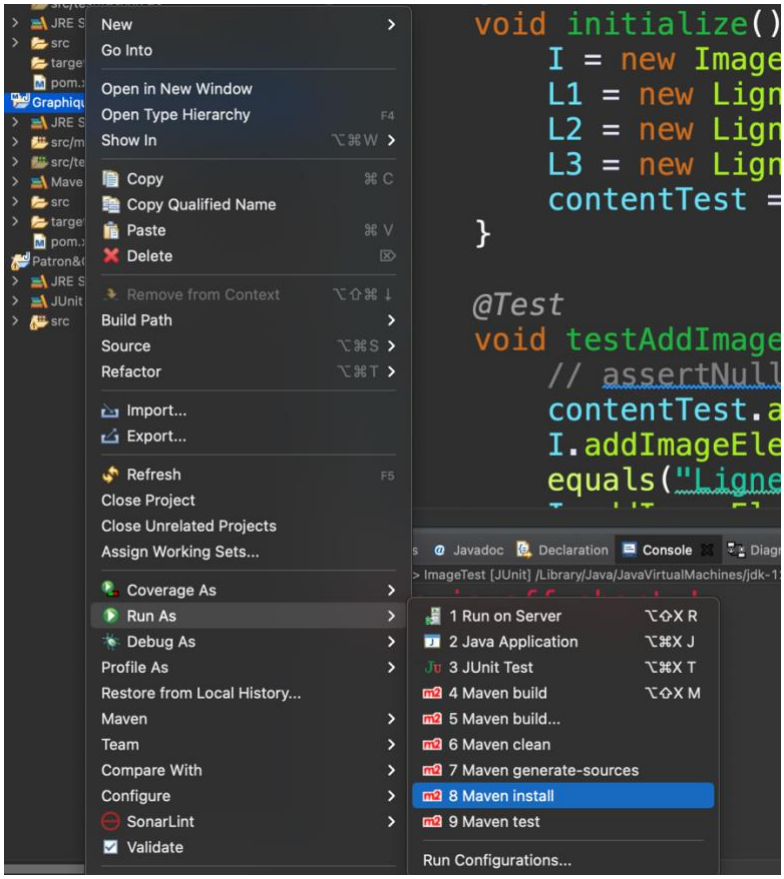

Figure 6 : Maven install

Pour nettoyer votre projet, une seule commande :
>mvn clean

Test qualité avec Maven

Le lancement d'un test qualité se fait maintenant à partir d'un goal maven :

```
mvn clean verify sonar:sonar -Dsonar.login=myAuthenticationToken
```

Attention, le mvn install doit passer avant de tester cette commande!

Vous pouvez aussi configure votre IDE préféré, ci-après l'exemple d'Eclipse (c'est une configuration que vous pouvez obtenir en faisant click-droit sur votre projet/Run As/ Run Configurations …/Maven build) :

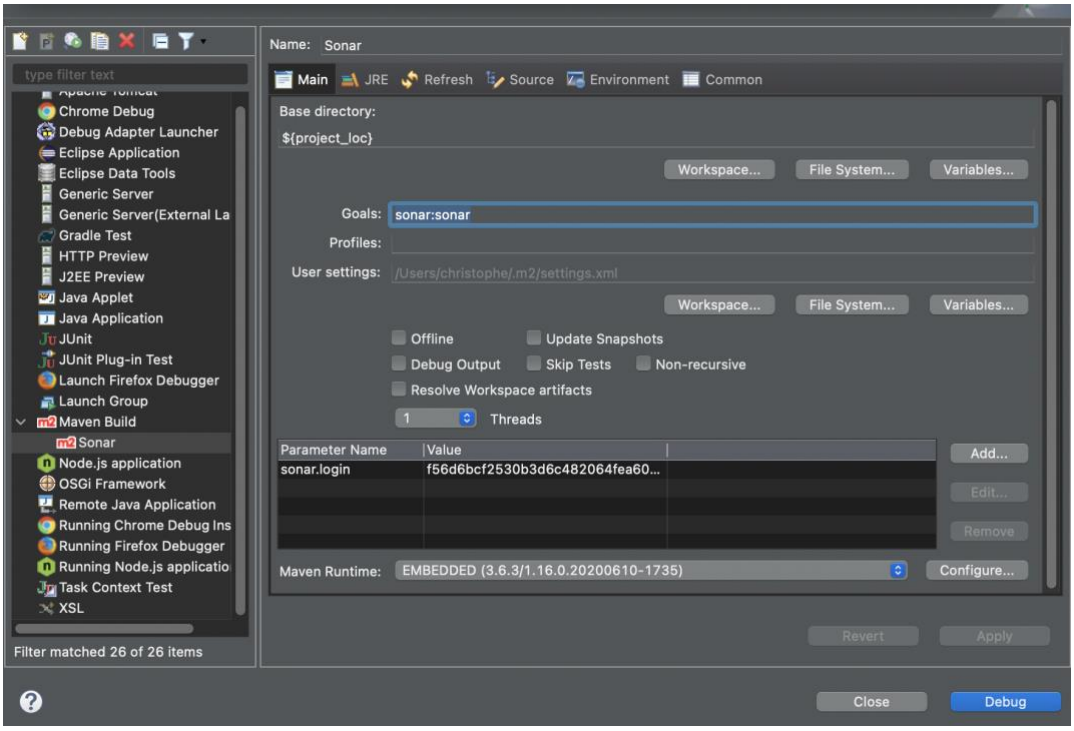

Figure 7 : Configuration Eclipse

Définissez bien le Base directory comme *$(project_loc)*, c'est ce qui vous premettra de lancer la configuration en ayant votre projet sélectionné.

Le résultat est visible immédiatement sur le serveur, vous pourrez surveiller entre autres :

- Votre couverture de tests
- Les codes smells dont Manuel vous a parlera !
- Les failles de sécurité

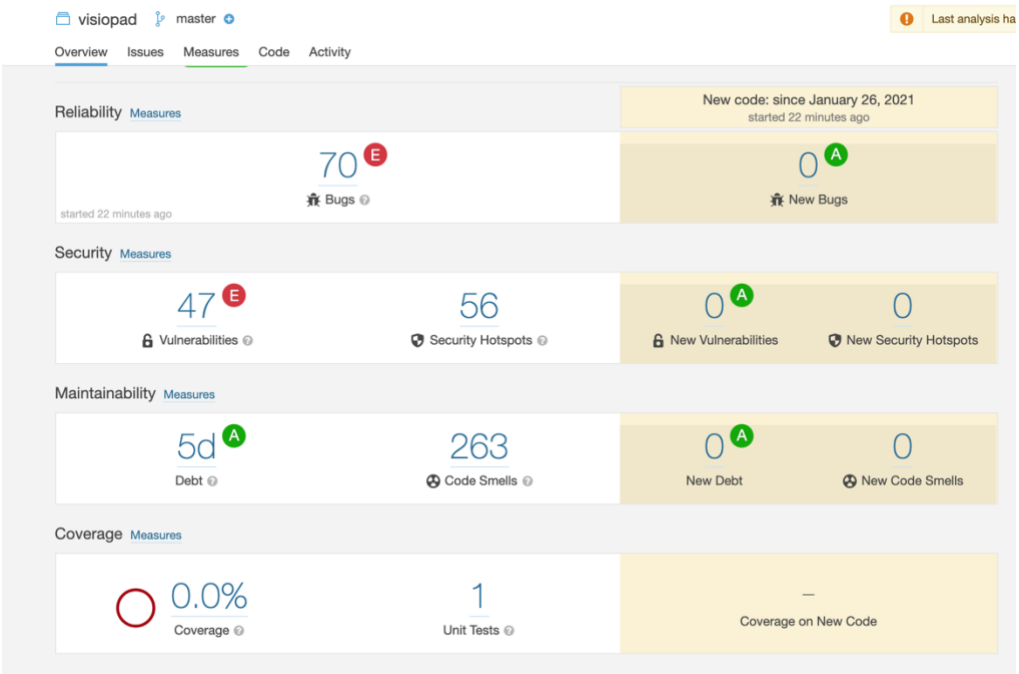

Figure 8 : Résultat sur SonarQube

Ces tests peuvent être réalisés de manière continue en les intégrant à un serveur Jenkins par exemple ou en les lançant manuellement, le serveur Sonar conservant les delta.

Test qualité avec Eclipse

SonarQube est le serveur qualité de vôtre projet. Pour visualiser en permanence les métriques de votre projet, installez SonarLint, un agent qui va se connecter directement au serveur et que vous pouvez trouver dans le MarketPlace Eclipse Help/Eclipse MarketPlace.

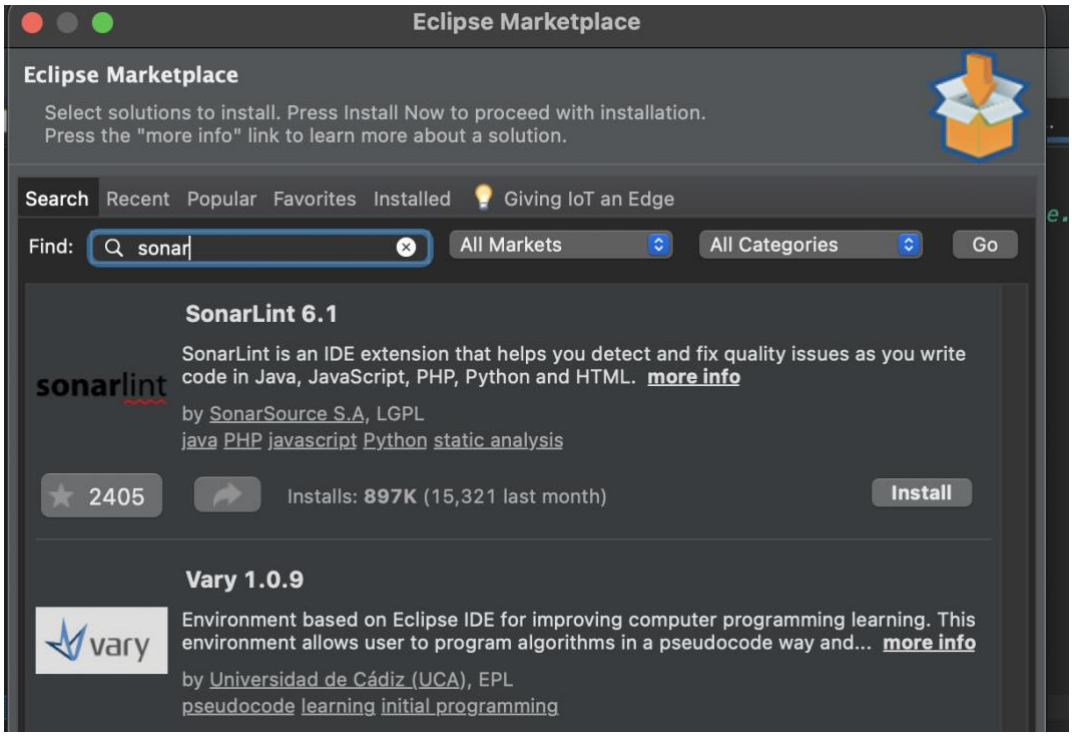

Figure 9 : Le marketplace Eclipse

Configurez le pour se connecter directement au serveur SonarQube.

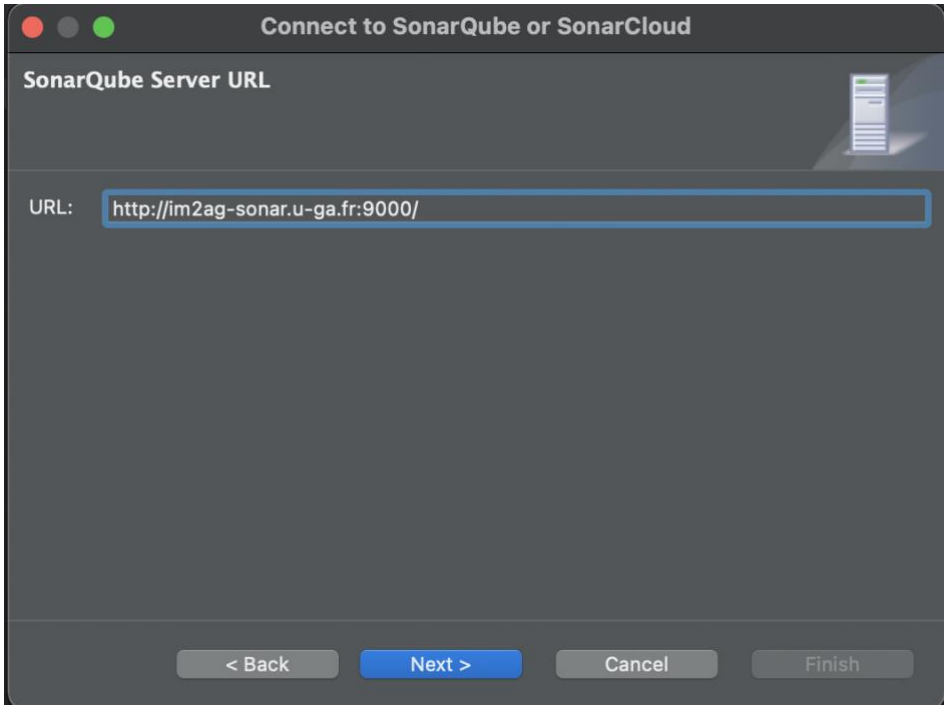

Figure 10 : Configuration de SonarLint

Enjoy !