Análisis de APIs con SonarQube

Este documento describe las distintas reglas disponibles para analizar APIs con SonarQube y su proceso de instalación

|  |  |  |
| --- | --- | --- |
| **Prepared by** | Deloitte | |
| **Consultant** | | Ángel Fernández Jabalera |
| **Senior Consultant** | | Cristina Sánchez Ruiz |

|  |  |
| --- | --- |
| **Version:** | 1.1 |
| **Date:** | 07-10-2020 |

**Control de versiones**

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Actualizado por | Descripción |
| 1.0 | 01/10/2020 | Ángel | Versión inicial |
| 1.1 | 07/10/2020 | Ángel | Se añade información sobre Quality profile y las modificaciones que realiza la instalación del plugin Mulesoft Catalyst Sonar en SonarQube. |
|  |  |  |  |
|  |  |  |  |

Contenido

[**1.** **Introducción** 4](#_Toc52983925)

[**2.** **Instalación de SonarQube** 5](#_Toc52983926)

[**2.1.** **Instalación SonarQube server** 5](#_Toc52983927)

[**2.2.** **Instalación Mulesoft Catalyst Sonar Plugin** 5](#_Toc52983928)

[**2.3.** **Instalación SonarQube Plugin en Anypoint Studio (Opcional)** 5](#_Toc52983929)

[**3.** **SonarQube Server** 7](#_Toc52983930)

[**3.1.** **Lenguajes** 7](#_Toc52983931)

[**3.2.** **Quality profiles** 7](#_Toc52983932)

[**3.3.** **Tipos de reglas** 9](#_Toc52983933)

[**3.3.1.** **Bugs** 9](#_Toc52983934)

[**3.3.2.** **Vulnerabilities** 10](#_Toc52983935)

[**3.3.3.** **Code Smell** 11](#_Toc52983936)

[**3.3.4.** **Security Hotspot** 12](#_Toc52983937)

[**3.3.5.** **Custom Rules** 12](#_Toc52983938)

# **Introducción**

El presente documento tiene como objetivos:

* Indicar los pasos seguidos para llevar a cabo la instalación de las herramientas necesarias para el correcto desarrollo de esta prueba de concepto.
* Revisar las distintas reglas ofrecidas por Mulesoft Catalyst para el análisis de APIs Mule.

# **Instalación de SonarQube**

SonarQube es una plataforma de código abierto utilizada para analizar la calidad del código en base a una serie de reglas definidas previamente. Esta herramienta resulta esencial en fases de pruebas puesto que, de forma rápida y estandarizada, puede analizarse el proyecto completo. La instalación de SonarQube para analizar APIs de Mulesoft cuenta con dos fases diferenciadas:

# **Instalación SonarQube server**

La instalación de SonarQube Server proporcionará el entorno local para realizar los distintos análisis y definir las reglas para estos análisis. Los enlaces de interés para esta instalación son:

* Descarga de SonarQube: <https://www.sonarqube.org/downloads/>
* Manual de instalación en Linux: <https://www.imaginaformacion.com/tutorial/como-instalar-sonarqube-en-linux/>

# **Instalación Mulesoft Catalyst Sonar Plugin**

La instalación de este plugin en SonarQube es necesario para añadir Mule como lenguaje disponible en el análisis. Su instalación no afecta a los proyectos existentes en SonarQube, ya que, su única funcionalidad es habilitar Mule como lenguaje disponible para el análisis y añadir sus reglas por defecto. Los enlaces de interés para esta instalación son:

* Descarga del plugin: <https://github.com/mulesoft-catalyst/mule-sonarqube-plugin>
* Manual de instalación del plugin: <https://dzone.com/articles/publish-mule4-code-review-and-coverage-to-sonarqub>

Para evitar errores asociados a la instalación de este plugin es necesario realizar la siguiente secuencia de acciones:

* Iniciar SonarQube Server
* Generar el fichero .jar siguiendo los pasos del manual de instalación e introducirlo en la ruta *sonar-home/extensions/plugins*
* Reiniciar SonarQube Server con el comando */etc/init.d/sonar restart*
* Introducir los ficheros de reglas para Mule 3 y/o Mule 4 en función de las necesidades en la ruta *sonar-home/extensions/plugins*
* Reiniciar de nuevo SonarQube Server con el comando */etc/init.d/sonar restart*
* Tras esto, el plugin quedaría instalado en SonarQube Server

Comando para ejecutar análisis:

mvn sonar:sonar

-Dsonar.projectKey=projectname

-Dsonar.host.url=http://localhost:9000

-Dsonar.login=tokenLogin

-Dsonar.sources=src/main,pom.xml

# **Instalación SonarQube Plugin en Anypoint Studio (Opcional)**

La instalación de este plugin permite enlazar los proyectos directamente desde Anypoint Studio con SonarQube Server. Los enlaces de interés para esta instalación son:

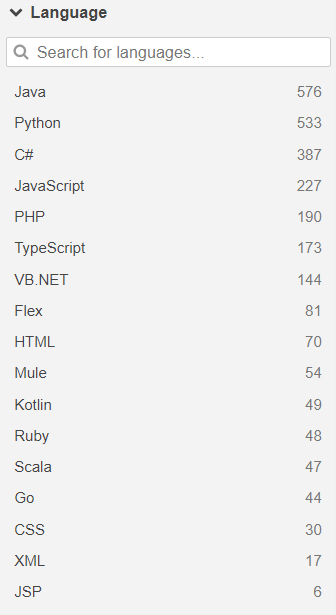
* Descarga del plugin: <http://downloads.sonarsource.com/eclipse/eclipse/>
* Manual de instalación del plugin: <http://mulesoft.github.io/connector-certification-docs/advanced/index.html#_sonarqube_eclipse_plugin_setup>

# **SonarQube Server**

SonarQube Server es el entorno en el cual se llevará a cabo el análisis de código. Antes de realizar el análisis, es necesario tener en cuenta varios factores. En primer lugar, se debe crear un nuevo proyecto y, tras eso, definir el lenguaje que queremos aplicarle. Una vez realizado el paso anterior, es necesario definir un “Quality Profile”, el cual nos permitirá definir cuáles son las reglas que se van a aplicar al análisis.

# **Lenguajes**

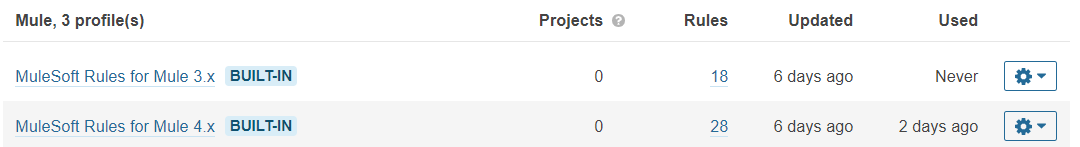
SonarQube es capaz de realizar análisis de código para distintos lenguajes. A continuación, se indica la lista de lenguajes soportados:



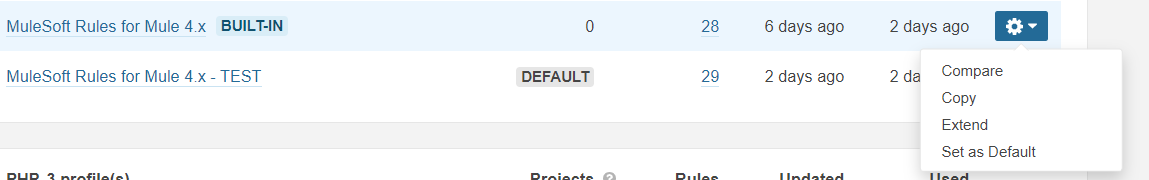
El número que aparece a la derecha indica la cantidad de reglas disponibles por defecto para cada lenguaje. En este documento se hace referencia al análisis mediante ***Mule Language***, el cuál ha sido añadido a la lista tras la instalación del plugin Mulesoft Catalyst en SonarQube Server.

# **Quality profiles**

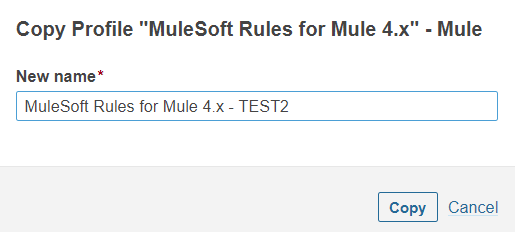
Para cada lenguaje encontramos definidos por defecto una serie de Quality profiles con el conjunto mínimo de reglas que representan buenas prácticas. Para el caso de Mule aparecen dos:



Para crear una nueva Quality profile se debe copiar una de las existentes:



Introducir el nombre deseado:



Y, a continuación, definir las reglas que necesitamos que estén activas:



Una vez creada la nueva Quality profile, se puede definir como Default para este tipo de análisis, es decir, será el conjunto de reglas a aplicar por defecto cuando se analice cualquier API Mulesoft. La creación de estas Quality profile nos permite establecer las reglas que se van a validar en cada análisis, siendo, por tanto, algo que no afecta a la configuración establecida para otros análisis en lenguajes distintos a Mule.

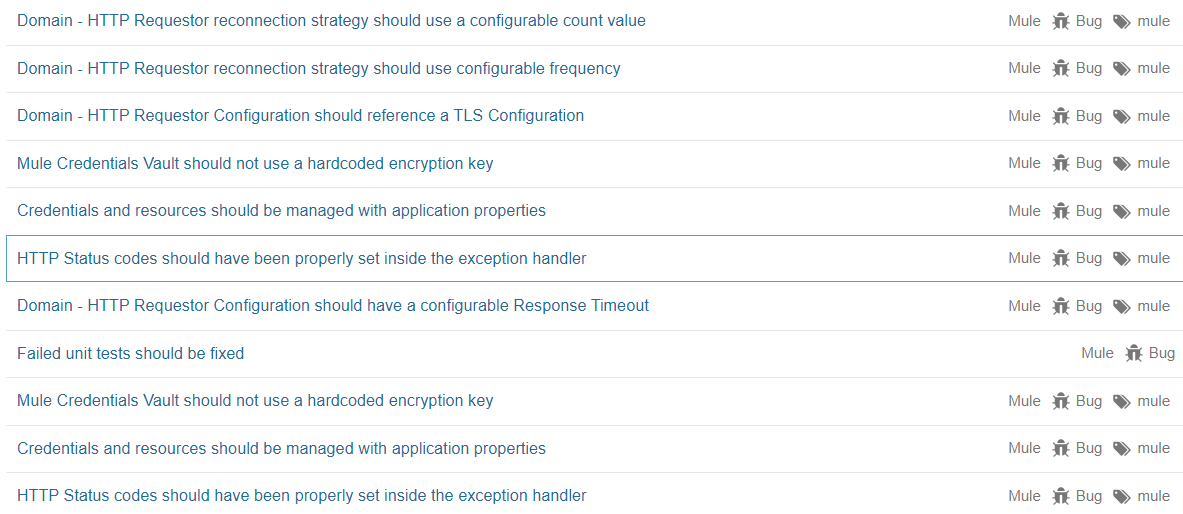
# **Tipos de reglas**

SonarQube diferencia entre los siguientes tipos de reglas:

* Bugs
* Vulnerabilities
* Code Smells
* Security Hotspots
* Custom Rules

# **Bugs**

Las reglas asociadas al tipo Bug que aparecen por defecto son:



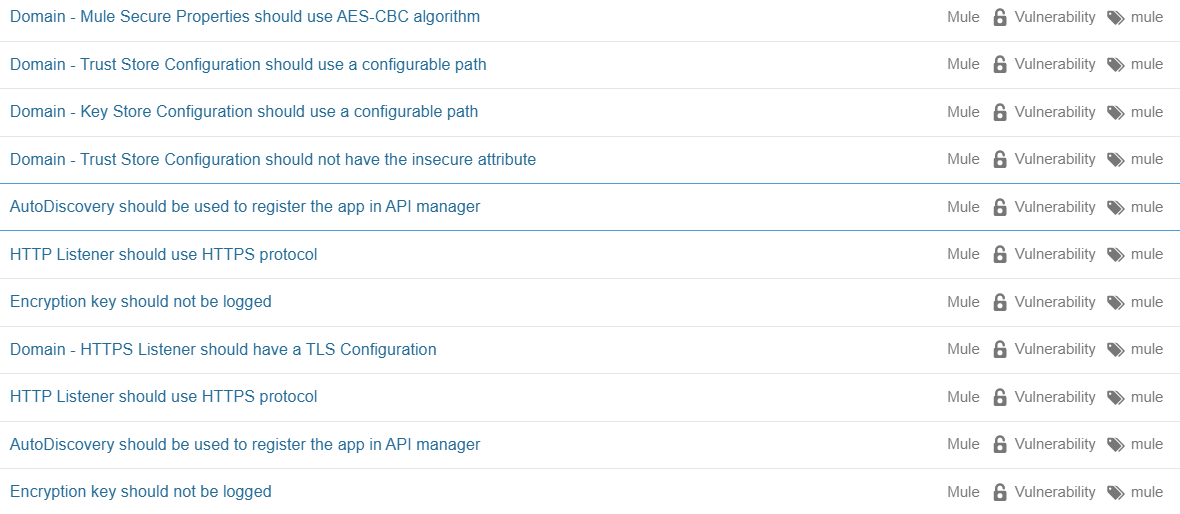
**Ejemplo**: *HTTP Status codes should have been propertly set inside the exception handler*

Alerta generada por SonarQube cuando no se controlan de forma correcta todos los errores asociados a un HTTP Request en un componente Error Handler:



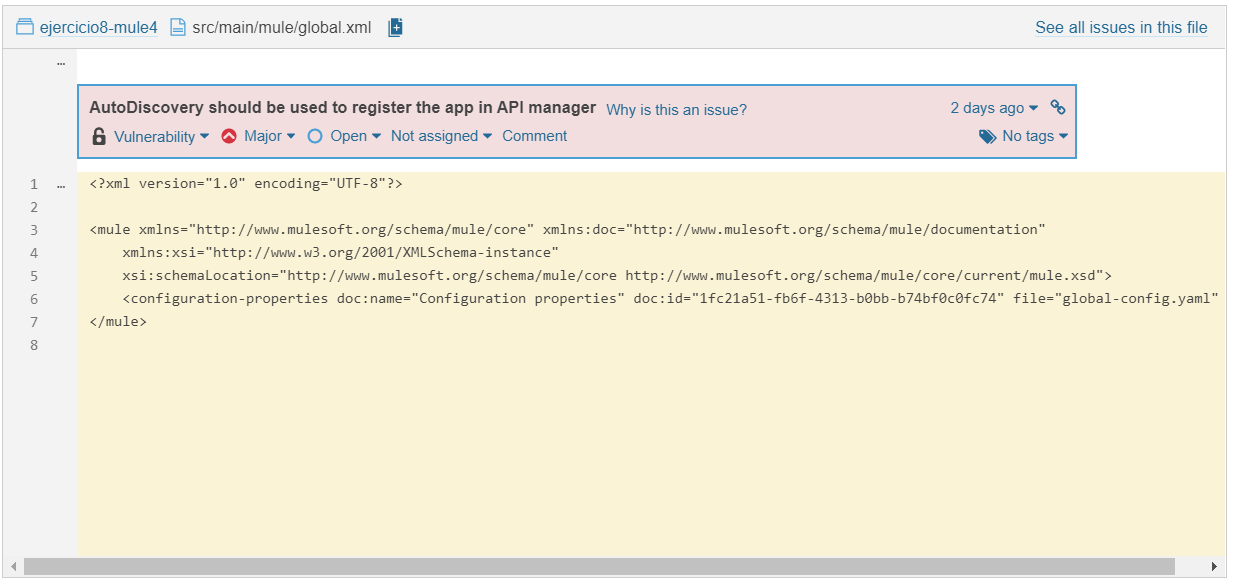
# **Vulnerabilities**

Las reglas asociadas al tipo Vulnerabilities que aparecen por defecto son:



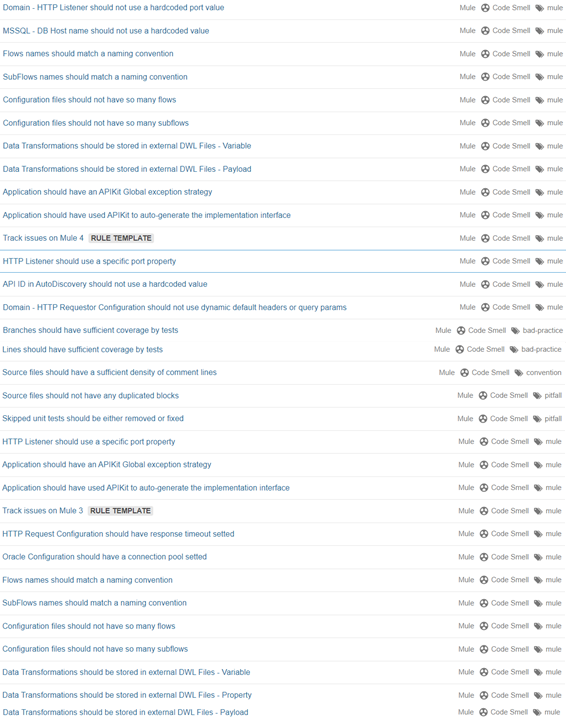
**Ejemplo**: *AutoDiscovery should be used to register the app in API manager*

Alerta generada por SonarQube cuando no se ha definido el componente AutoDiscovery para realizar la conexión adecuada con API Manager de Anypoint Platform:



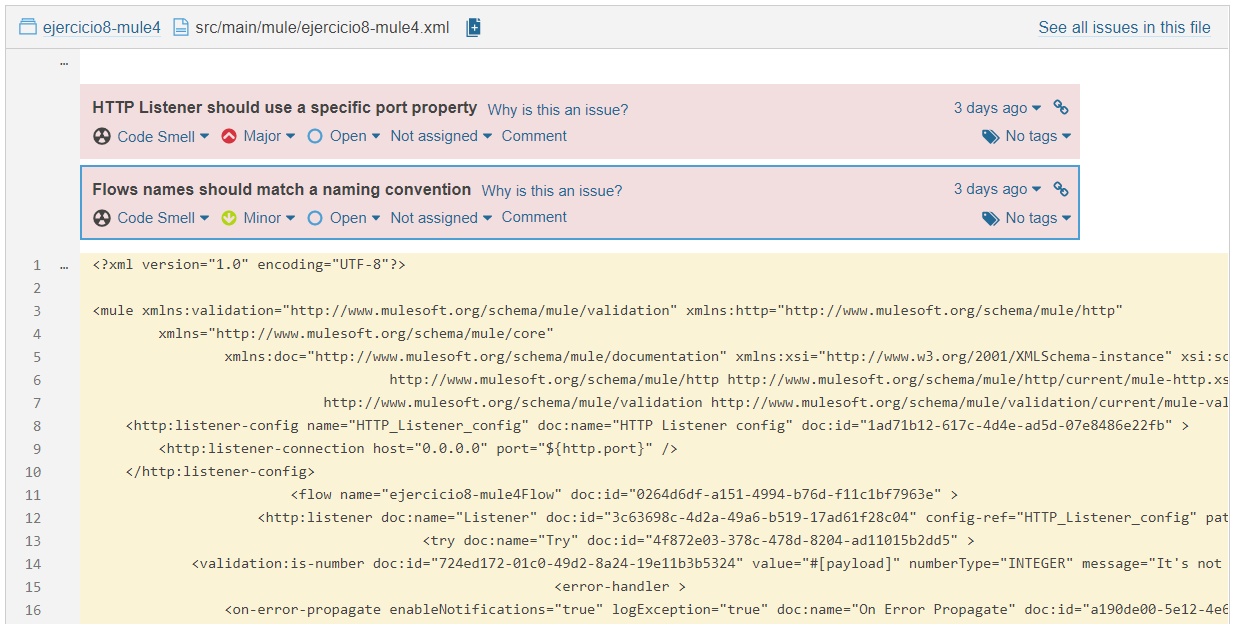
# **Code Smell**

Las reglas asociadas al tipo Code Smell que aparecen por defecto son:



**Ejemplo**: *Flows names should match a naming convention*

Alerta generada por SonarQube cuando los distintos flujos presentes en la API no siguen una nomenclatura definida para el nombre del flujo:



# **Security Hotspot**

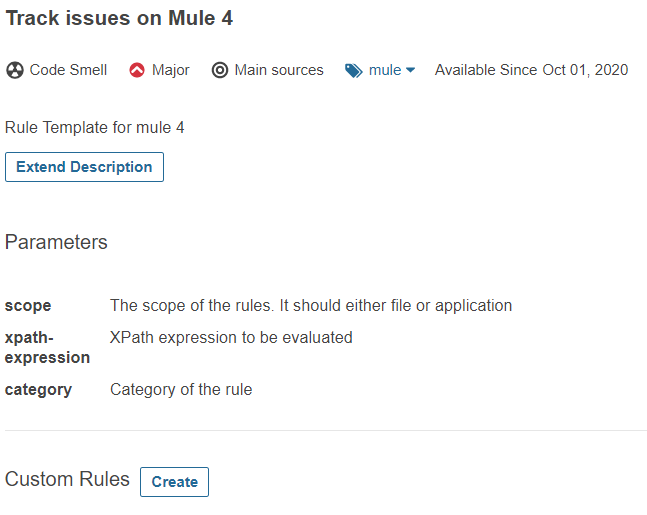
Las reglas asociadas a este tipo de alerta no aplican para el lenguaje Mule en SonarQube.

# **Custom Rules**

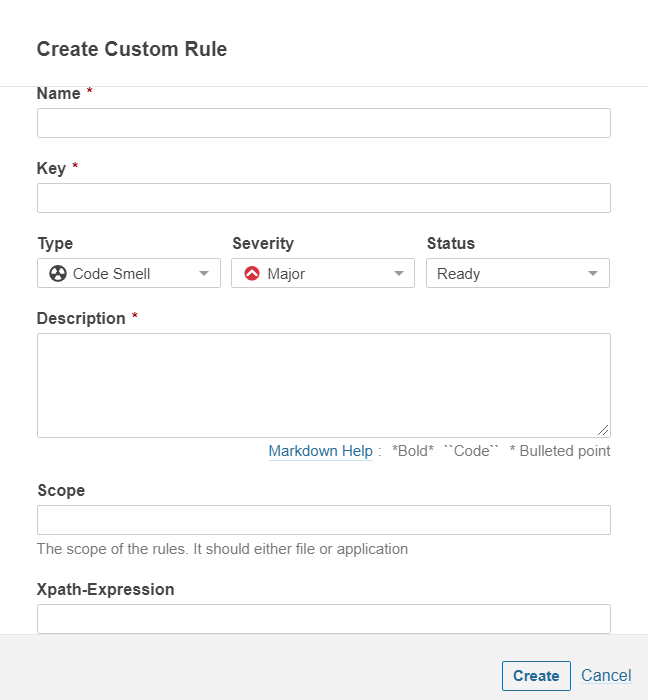
Además de las reglas definidas por defecto en SonarQube para Mule, es posible crear nuevas reglas personalizadas en nuestros análisis. Para crear estás nuevas reglas, es necesario hacer click en el siguiente elemento disponible en el listado de reglas:



A continuación, hay que hacer click en el botón marcado:



Y rellenar el cuadro con la información necesaria para aplicar la regla:



Finalmente es necesario añadir la regla en el “Quality Profile” deseado para realizar el análisis.

**Ejemplo**: *HTTP Request should have untill success*

Alerta generada por SonarQube cuando el número de HTTP Request es diferente al número de componentes Until Successful dentro de la API:

