DOCUMENTO DEL AVANCE #2

para

Sistema de preprocesamiento y segmentación de células

Versión 1.0

Preparado por:

José Alvarado Chaves, 201129079
Reggie Barker Guillén, 2014050578
Joel Barrantes Garro, 2013120962
Joel Schuster Valverde, 2014096796

Instituto Tecnológico de Costa Rica

28 de abril de 2017

Índice general

1 Unidades de diseño

En este capitulo, se relacionan los ítems de diseño identificados anteriormente con los requerimientos y los artefactos de la arquitectura.

1.1. Validación del sistema

En la siguiente tabla se muestra cada item de diseño asociado y su correspondencia con los requerimientos descritos en la Especificación de Requerimientos del Sistema(SyRS):

Número de item	Nombre de item	Requerimiento asociado
1	Diagrama de clases del sistema.	Procesamiento de lotes de imágenes en el sistema;
2	Diagrama de estados de procesamiento de un lote.	Subida de imágenes; Procesamiento de lotes de imágenes en el sistema; Notificación de procesamiento del lote
3	Diagrama de componentes del sistema.	Pre-procesamiento de lotes de imágenes en el sistema;
4	Diagrama de despliegue del sistema.	Subida de imágenes; Descarga de imágenes procesadas del lote
5	Diagrama de actividad de la interacción entre usuario y sistema.	Manejo de sesiones de usuario; Visualización de lotes pendientes; Descarga de imágenes procesadas del lote; Notificación de procesamiento del lote
6	Diagrama de casos de uso del investigador.	Visualización de lotes pendientes; Descarga de imágenes procesadas del lote; Informe de resultados del procesamiento de un lote
7	Diagrama de Entidad-Relación.	Manejo de sesiones de usuario;
8	Diagrama de mongoDB	Subida de imágenes; Descarga de imágenes procesadas del lote;

1.2. Validación de la implementación

La siguiente tabla muestra relación entre los items de implementación con los items de diseño del sistema:

Número de item	Nombre de item	Item de implementación
1	Diagrama de clases del sistema.	Clases del *package* tec.psa;
2	Diagrama de estados de procesamiento de un lote.	Imagen.java; Lote.java
3	Diagrama de componentes del sistema.	Spring Framework;
4	Diagrama de despliegue del sistema.	Spring Framework;
5	Diagrama de actividad de la interacción entre usuario y sistema.	Spring MVC; dashboard.html; login.html; registration.html; upload.html
6	Diagrama de casos de uso del investigador.	Spring MVC; dashboard.html; login.html; registration.html; upload.html
7	Diagrama de Entidad-Relación.	Clases del *package* tec.psa.model; tablas.sql
8	Diagrama de mongoDB	ImageManagement.java; mongoScript.bson

2 Estándares

2.0.1. Estándar de codificación

Durante el desarrollo del sistema SPACE, el equipo seguirá el estándar de codificación de Google, conocido como el *Google Java Style* para la codificación en el lenguaje de programación *Java*. Este estándar, junto al de Sun, es uno de los más populares entre los desarrolladores de Java, y sus lineamientos actualizados pueden encontrarse fácilmente en la web: https://google.github.io/styleguide/javaguide.html

Entre las razones para favorecer a este estándar de codificación, están

a. El equipo de desarrollo ya se encuentra familiarizado con el estándar;

b. Es un estándar actualizado y más moderno en comparación al que ofrece Sun;

c. Es de fácil acceso y de rápida lectura y comprensión;

d. Los lineamientos del estándar son sencillos pero completos, y permiten codificar a un buen ritmo, ideal para una metodología ágil;

e. Existen herramientas de verificación de estándar de calidad que incorporan este estándar por defecto;

f. Y existen herramientas en Eclipse que incorporan este estándar fácilmente en el entorno de programación.

2.1. Herramienta de verificación

Como herramienta de verificación del estándar, se utilizará el entorno de programación Eclipse Neon con la extensión Checkstyle, configurada para verificar el cumplimiento del *Google Java Style*. La herramienta puede instalarse fácilmente en Eclipse y es de código abierto. Se utilizará la versión 7.6.0 de Checkstyle. El sitio web oficial de la herramienta está en la dirección URL http://eclipse-cs.sourceforge.net. El código fuente de la herramienta también está disponible en SourceForge para su inspección.

Los pasos necesarios para la instalación de esta herramienta se muestran a continuación:

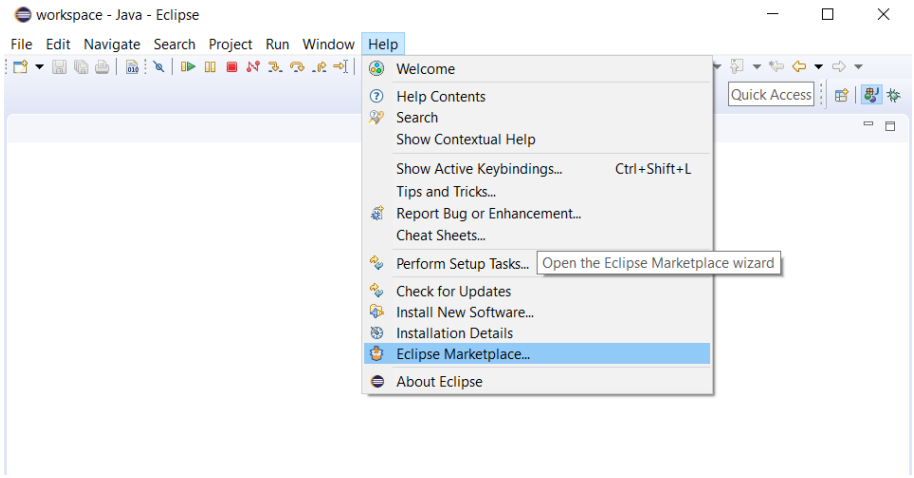

Figura 2.1: Marketplace de Eclipse

Desde el menú de ayuda de Eclipse, se accede al *Marketplace*.

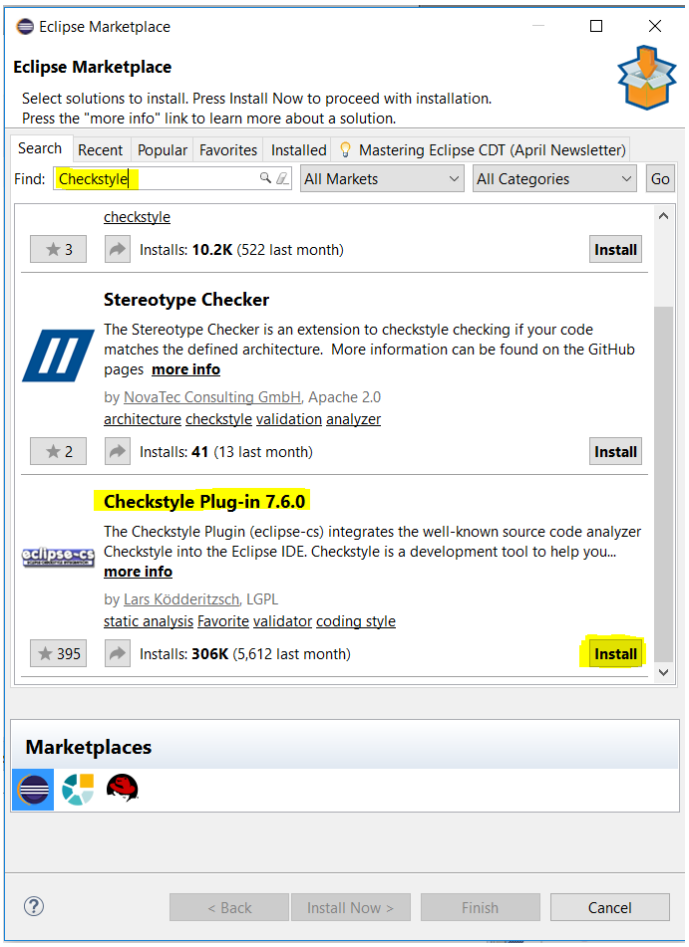

Figura 2.2: Plug-in de Checkstyle

Se busca por la herramienta *Checkstyle* en la barra de búsqueda. El complemento llamado *Checkstyle Plug-in 7.6.0* debería aparecer como resultado de la búsqueda. Se presiona

el botón *Install*.

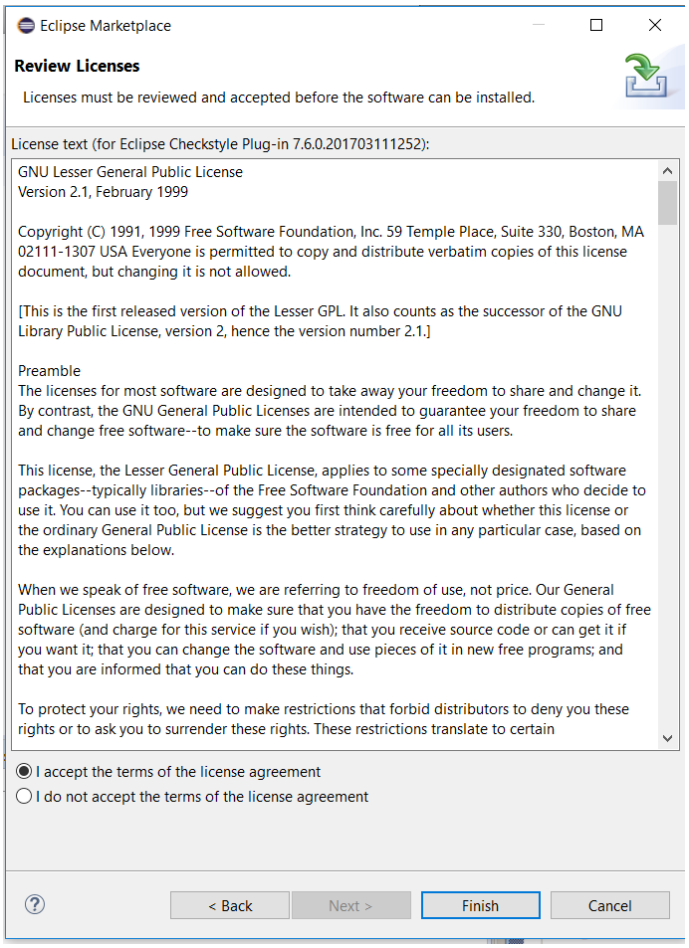

Figura 2.3: Instalación del Plug-in

Instalamos la herramienta siguiendo los pasos solicitados por el *Marketplace*.

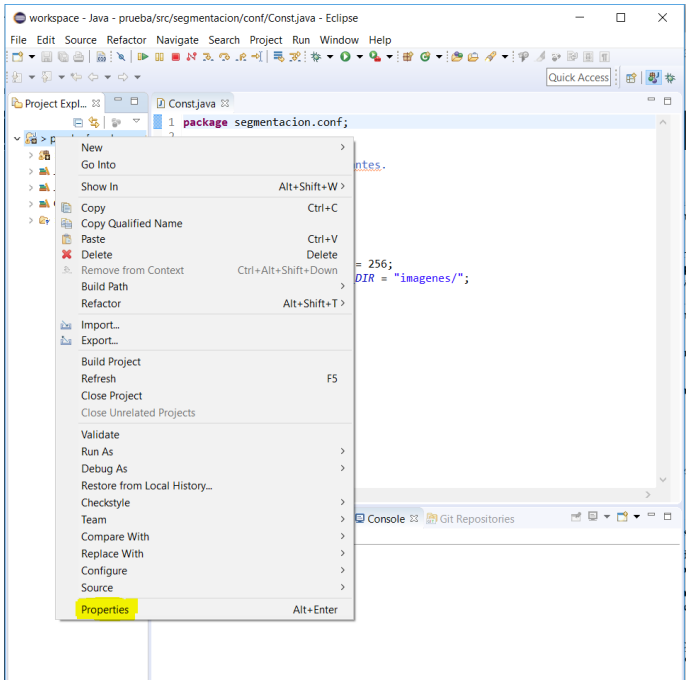

Figura 2.4: Configuración del proyecto

Una vez instalado el complemento, abrimos la ventana de propiedades del proyecto.

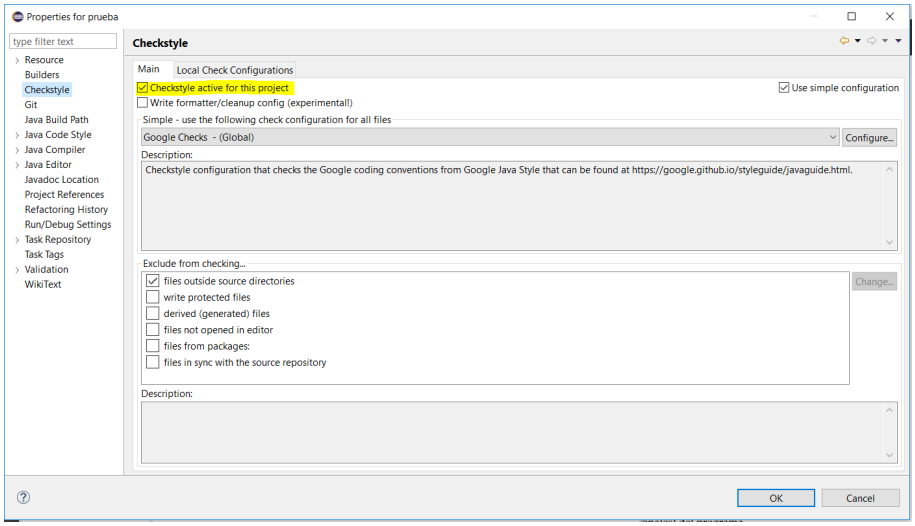

Figura 2.5: Habilitar Plug-in

Una vez en la ventana de propiedades, vamos a la pestaña *Checkstyle* y marcamos la opción *Checkstyle active for this project*. En esta misma pentaña podemos seleccionar el estándar de codificación(En nuestro caso, Google Java Style)

2.2. Jenkins (Puntos extra)

3 Repositorio

El repositorio del sistema *SPACE* (github.com/jubileus95/SPACE) contiene todos los ítems de configuración del proyecto. A continuación, se muestra el procedimiento para obtener la última versión de SPACE.

3.1. Obtener versión actual del sistema

Para obtener la línea base del sistema, se puede hacer un git clone del repositorio donde esta alojado el proyecto SPACE.

```
~\Documents> git clone https://github.com/jubileus95/SPACE.git
Cloning into 'SPACE'...
remote: Counting objects: 361, done.
remote: Compressing objects: 100% (249/249), done.
remote: Total 361 (delta 73), reused 287 (delta 54), pack-reused 0
Receiving objects: 100% (361/361), 1.12 MiB | 23.00 KiB/s, done.

Resolving deltas: 100% (73/73), done.
```

Figura 3.1: Git clone del proyecto

El URL para clonar el repositorio es el siguiente: https://github.com/jubileus95/SPACE.git

4 Métricas

Para el proyecto de SPACE, se van a emplear varias métricas para tener un mejor manejo de implementación, compreción y mantenibilidad del sotfware. Se mostraran dos métricas para los requerimientos, una para los funcionales y otra para los no funcionales.

4.1. Verificación de olores de software

Para encontrar bugs y olores de software en el código estático directamente desde el entorno de programación de Eclipse, se puede utilizar la herramienta *SonarLint for Eclipse* de SonarSource (desarrolladores además de SonarQube). Para instalar la herramienta en Eclipse, se debe ir al menú *Help*, seleccionar la opción *Eclipse Marketplace...* y buscar *SonarLint*, lo cual despliega la ventana flotante que se puede ver a continuación:

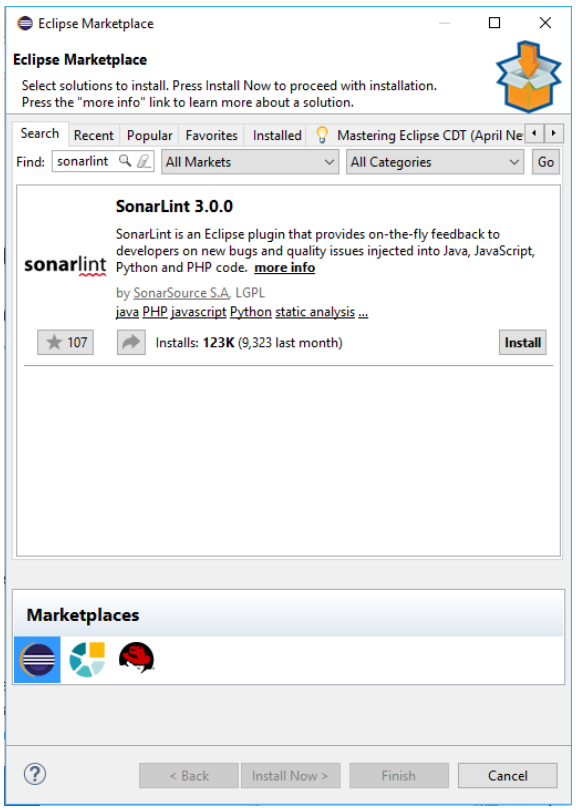

Figura 4.1: Instalación de SonarLint

La configuración de la herramienta es automática y no requiere de intervención significativa del usuario. Para acceder a las capacidades de reportes de SonarLint, en Eclipse se debe navegar a *Window* > *Show View* > *Other...* y una vez ahí, buscar la opción de *SonarLint Report*.

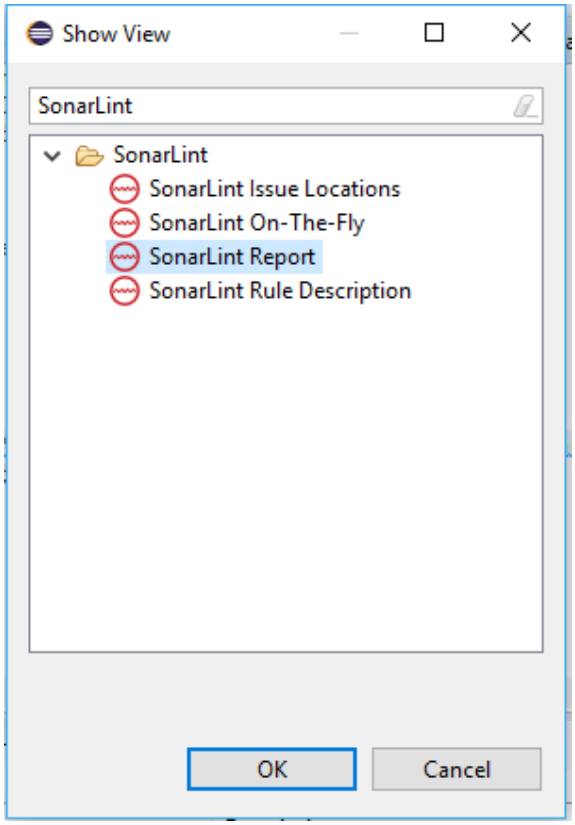

Figura 4.2: Abrir ventana de reportes de SonarLint

Para obtener un reporte completo de los bugs, olores de software y recomendaciones, basta con ejecutar el escáner sobre el proyecto, con el botón *Current project*. De forma automática, se muestran los resultados del análisis, como se aprecia en la siguiente figura.

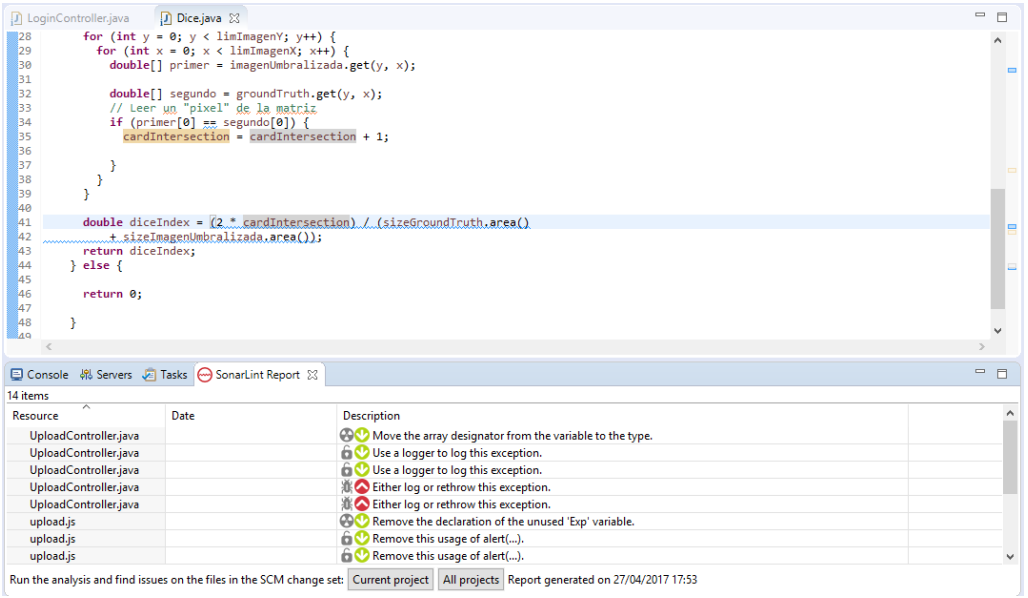

Figura 4.3: Reporte de SonarLint

4.2. Complejidad ciclomática

Para calcular la complejidad ciclomática en el código estático fácilmente desde el entorno de programación de Eclipse, se puede utilizar la herramienta *Codecity 0.9.0*. Para instalar la herramienta en Eclipse, se debe ir al menú *Help*, seleccionar la opción *Eclipse Marketplace...* y buscar *Codecity*, lo cual despliega la ventana flotante que se puede ver a continuación:

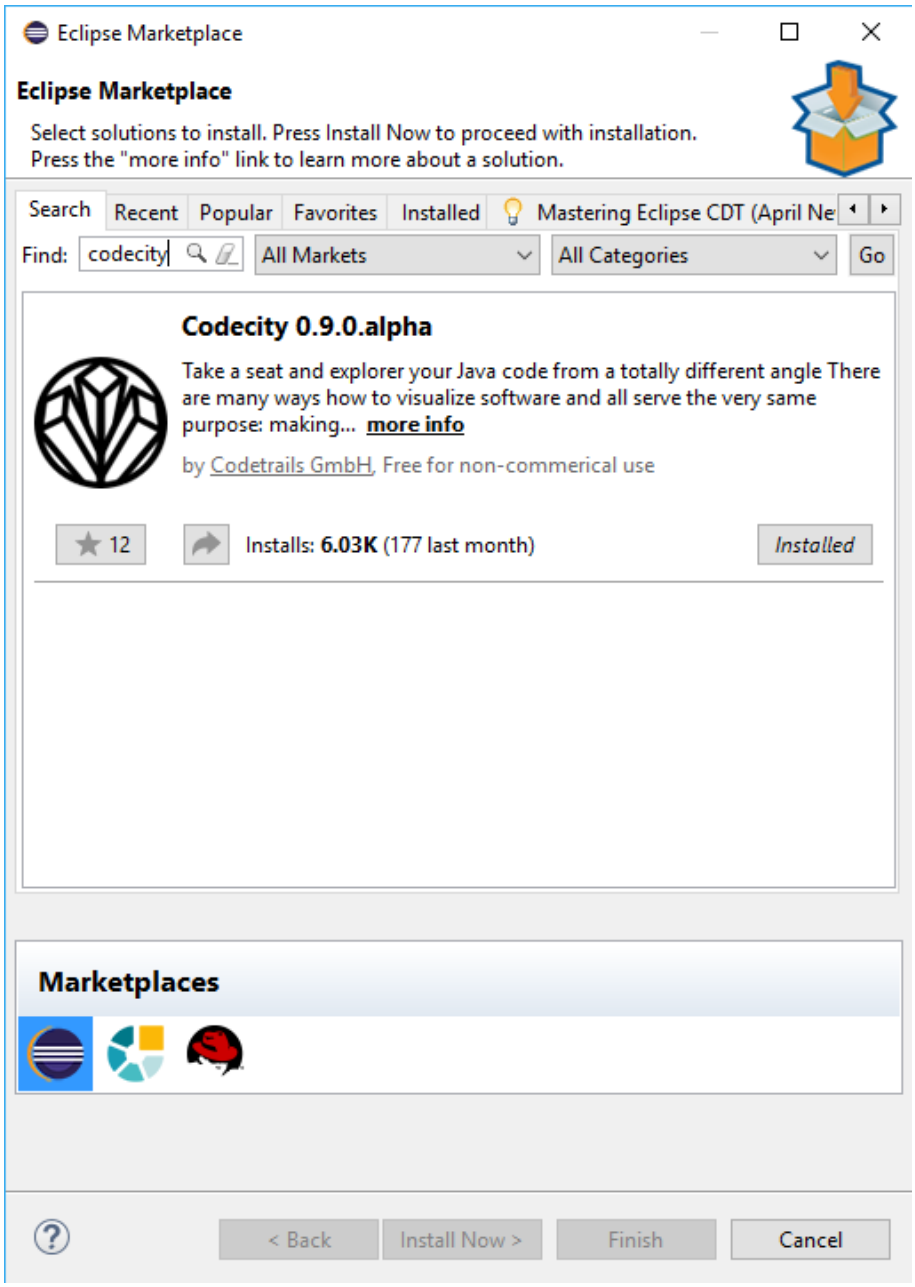

Figura 4.4: Instalación de Codecity

Esta herramienta tampoco requiere de más configuración para su utilización. Luego, para acceder a los reportes de Codecity, en Eclipse se debe hacer clic derecho sobre el proyecto que se desea analizar, y de ahí seleccionar *Show In* > *Codecity*, como se muestra en la siguiente imagen.

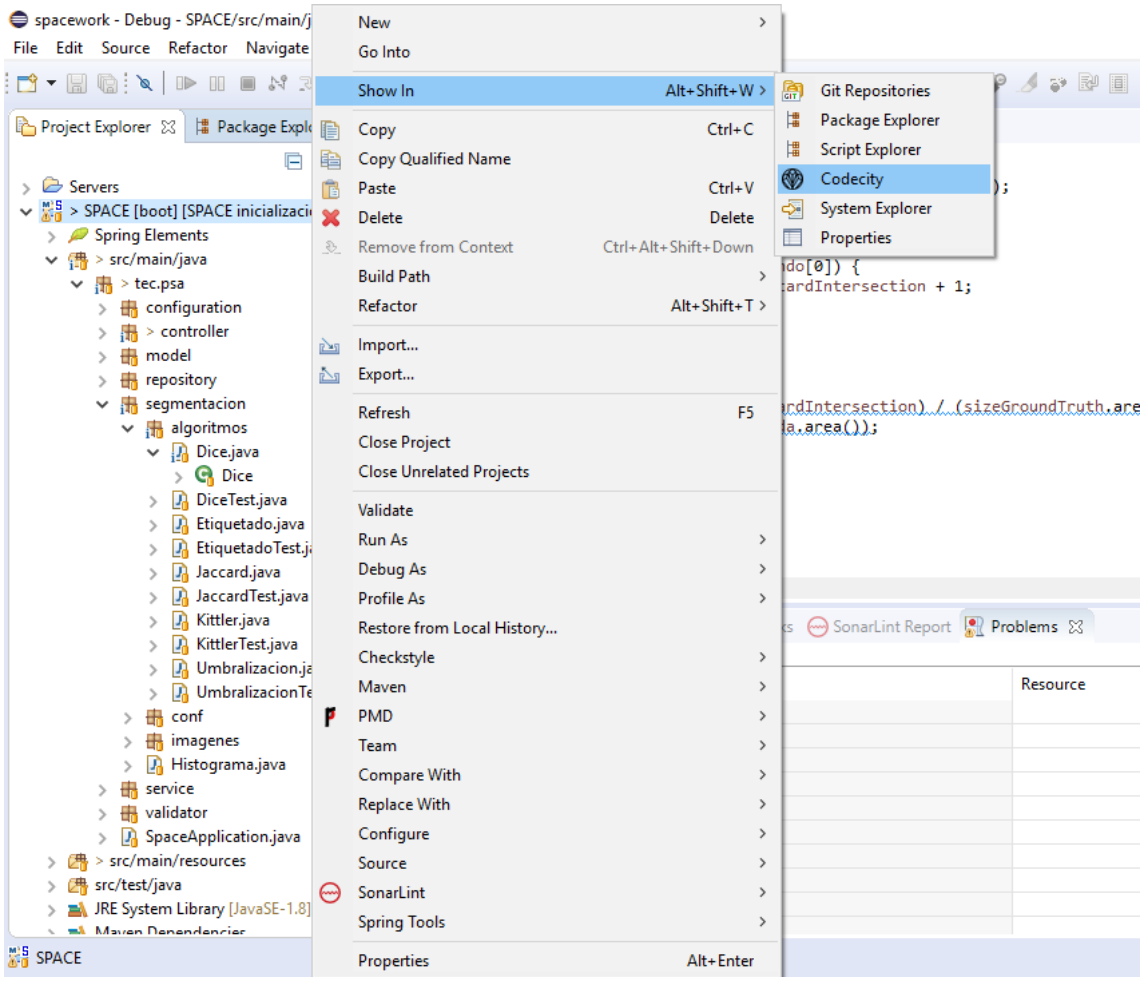

Figura 4.5: Solicitar reporte de Codecity

Finalmente, la herramienta redirige el flujo al navegador web para visualizar diferentes métricas, entre ellas la complejidad ciclomática. Basta con seleccionar *Cyclomatic complexity* como la métrica deseada. Opcionalmente, se pueden hacer ajustes sobre los colores y demás información mostrada en las dimensiones. Al pasar el puntero sobre el gráfico, la herramienta muestra más información, por ejemplo, el componente del sistema que genera el dato seleccionado en particular.

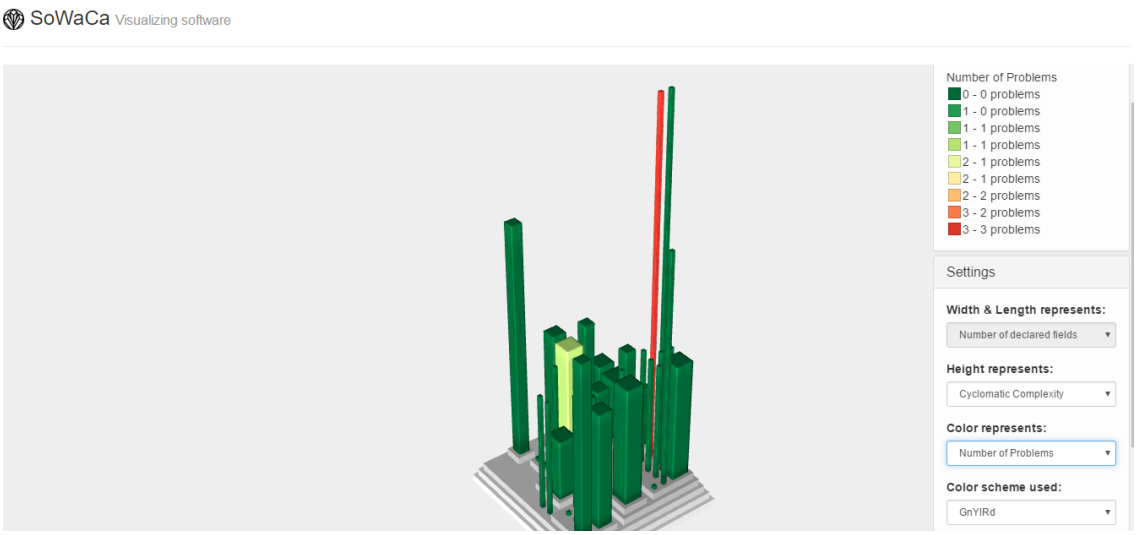

Figura 4.6: Gráfico de complejidad ciclomática en Codecity

Como puede apreciarse, existen dos barras verticales que destacan sobre las demás, que corresponden a dos archivos de código fuente del sistema que tienen una complejidad ciclomática de 6. No obstante, una complejidad ciclomática de hasta 10 es aceptable para la mantenibilidad del código.

4.3. Cambios en límites de métricas

A partir de la información adquirida anteriormente con las herramientas automáticas, se encontró que para los bugs y olores del código **(madurez)** se había definido un límite adecuado, ya que se propuso que no se dieran más de 10 olores ni más de 10 bugs en el análisis del código estático. Esto se cumplió en las métricas obtenidas.

Por otro lado, el límite se actualizó de un valor de 2 a 10 para la **complejidad ciclomática**, ya que se obtuvo una evaluación de 6 para un código simple. Tras hacer la investigación, se descubrió que una complejidad ciclomática de hasta 10 es aceptada como código sencillo de entender. Además, se cambió la herramienta *Metrics for Eclipse* por Los cambios sobre los límites fueron actualizados en el documento *T1 - Atributos de calidad y métricas*.

5 Especificación de pruebas unitarias

En este apartado, se muestran la especificación de las pruebas unitarias implementadas en esta etapa del proyecto.

5.1. Kittler

Objetivo	Implementación del algoritmo de Kittler determinar un valor óptimo con el fin de umbralizar una imagen.
Tipo de entrada	Histograma
Entrada esperada	cuadro_005.bmp
Entrada atípica	Ninguna (solo imágenes)
Tipo de salida	Int: $\tau \in \{0, 1, 2, \dots , 254, 255\}$
Salida esperada	166

5.2. Umbralización

Objetivo	Umbraliza una imagen en blanco y negro con un valor de umbral determinado por el algoritmo de Kittler.
Tipo de entrada	Histograma, Tao (Int)
Entrada esperada	cuadro_005.bmp
Entrada atípica	Ninguna (solo imágenes)
Tipo de salida	Mat
Salida esperada	Matriz umbralizada en blanco y negro

5.3. Etiquetado

Objetivo	Etiqueta cada una de las células en una imagen con una característica distinguible (color).
Tipo de entrada	Imagen (Mat)
Entrada esperada	cuadro_005.bmp
Entrada atípica	Ninguna (solo imágenes)
Tipo de salida	Mat
Salida esperada	Imagen etiquetada con colores secuenciales

5.4. Dice (nueva)

Objetivo	Calcular valor que permita medir qué tan similar es una imagen segmentada automáticamente respecto a un groundtruth.
Tipo de entrada	Imagen (Mat)
Entrada esperada	Dice.bmp; Dice2.bmp; Dice3.bmp; Dice4.bmp
Entrada atípica	Ninguna
Tipo de salida	double : [0, 1]
Salida esperada	0; 0.25; 0.50; 0.75

5.5. Jaccard (nueva)

Objetivo	Calcular valor que permita medir qué tan similar es una imagen segmentada automáticamente respecto a un groundtruth.
Tipo de entrada	Imagen (Mat)
Entrada esperada	Dice.bmp; Dice2.bmp; Dice3.bmp; Dice4.bmp
Entrada atípica	Ninguna
Tipo de salida	double : [0, 1]
Salida esperada	0; 0.1429; 0.3333; 0.60

5.6. Subida a MongoDB (nueva)

Objetivo	Implementación de la creación e insersión de imágen de la base datos.
Tipo de entrada	Image
Entrada esperada	imagenr
Entrada atípica	Ninguna (solo imágenes)
Tipo de salida	Boolean
Salida esperada	State

5.7. Descarga de MongoDB (nueva)

Objetivo	Implementación de la busqueda y descarga de imágen de la base datos.
Tipo de entrada	String
Entrada esperada	imagenr
Entrada atípica	Ninguna (solo Cadenas)
Tipo de salida	Image
Salida esperada	imagenr (Imagen)