Documento del sistema

Aplicaciones Industriales

Andrés Tena De Tena

Javier Albaráñez Martínez

Índice

[I. Introducción 1](#_Toc70790545)

[II. Despliegue 6](#_Toc70790546)

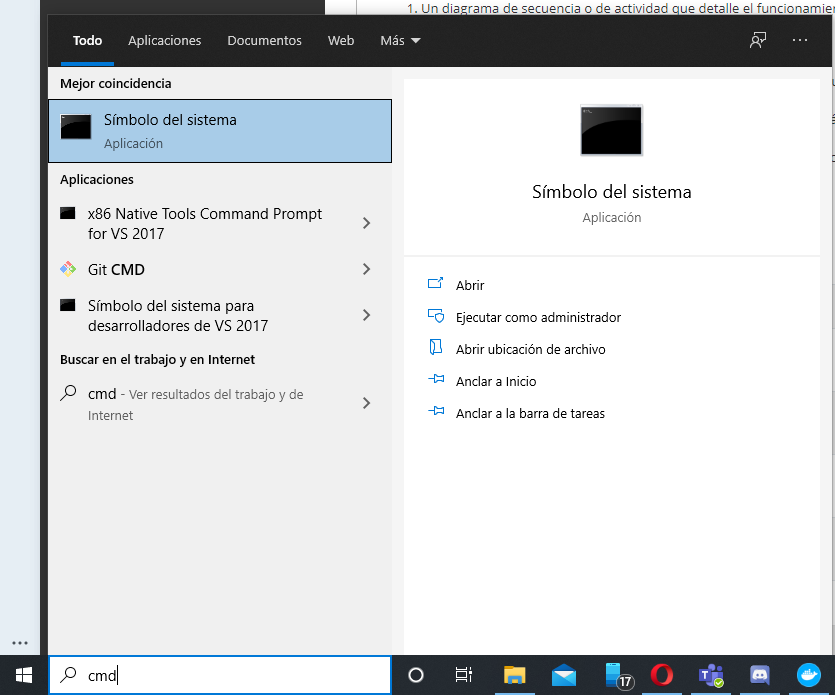
[III. Funcionamiento del sistema 7](#_Toc70790547)

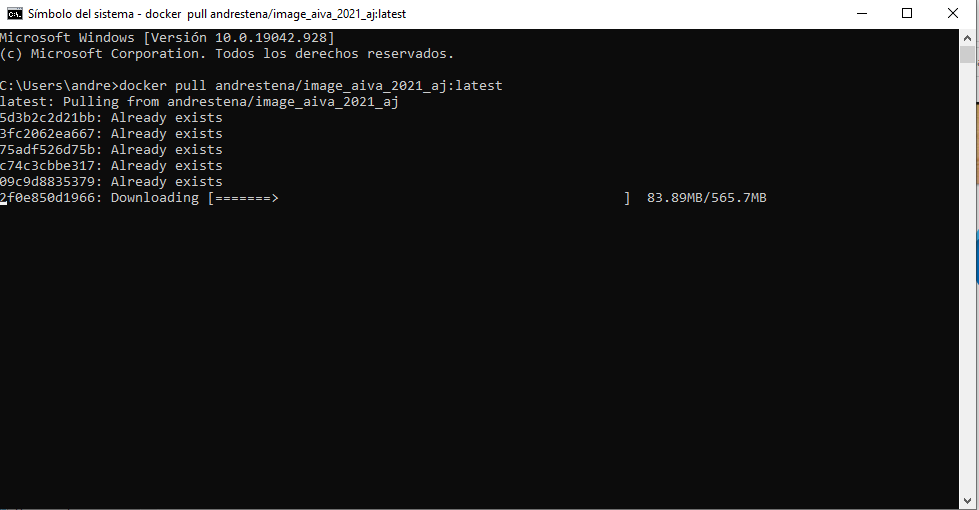
1. Introducción

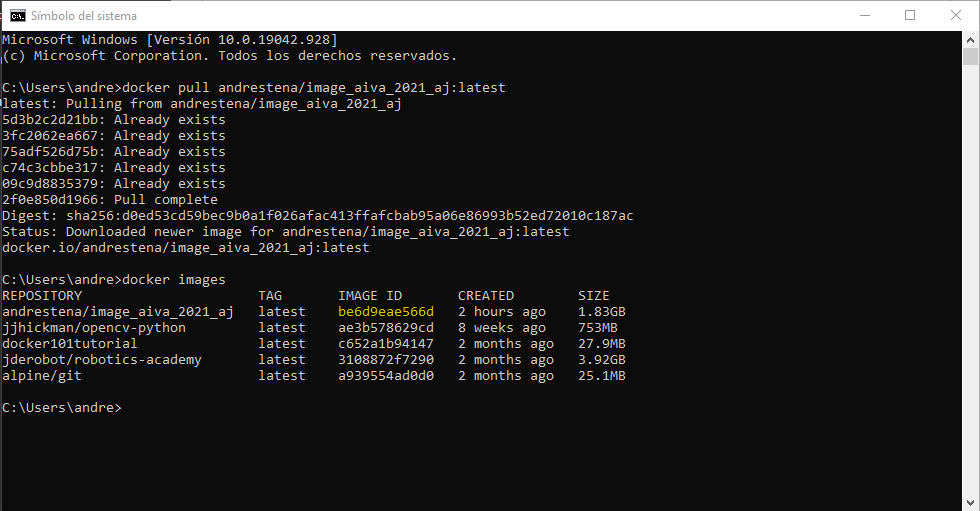
Se ha desarrollado una aplicación capaz de identificar y numerar el porcentaje de imperfecciones que posee una pieza de madera. Además, esta aplicación se ha subido a una imagen Docker desde la cual se puede ejecutar sin problema, simplemente ejecutando un archivo .java. A continuación, se detallan los pasos a seguir para instalar la imagen Docker en Windows y ejecutar la aplicación:

Enlace a GitHub:

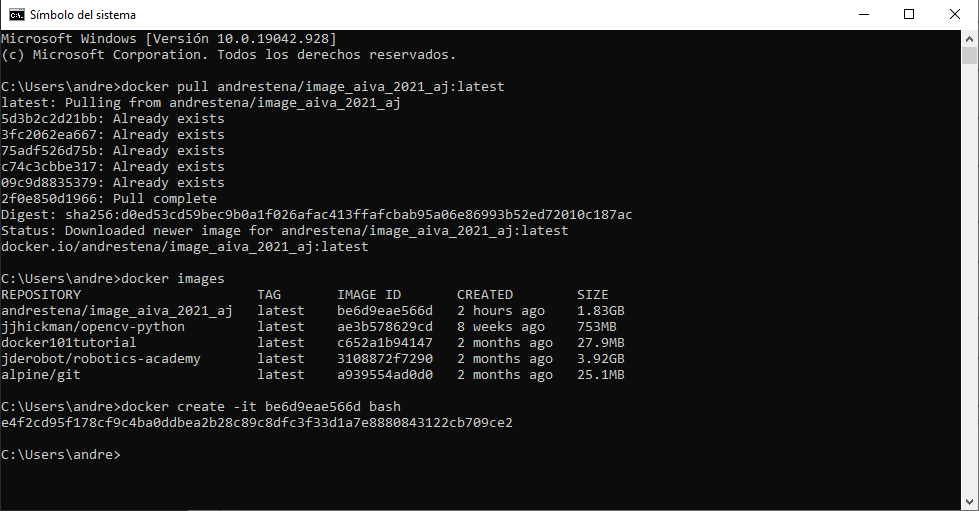
<https://github.com/AlbaranezJavier/AIVA_2021_AJ>

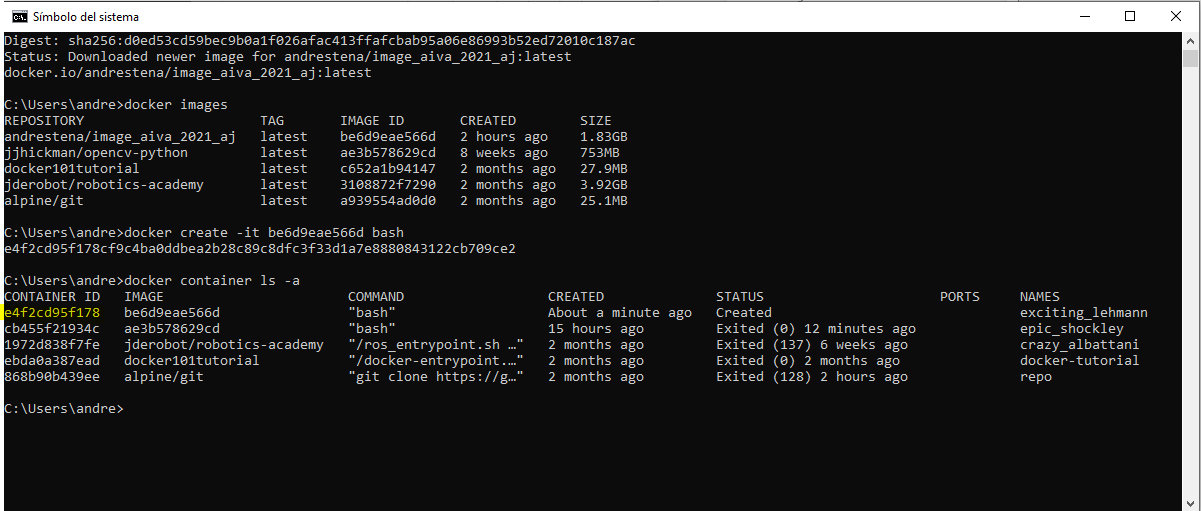
Paso 1: Abrir la cmd de Windows.

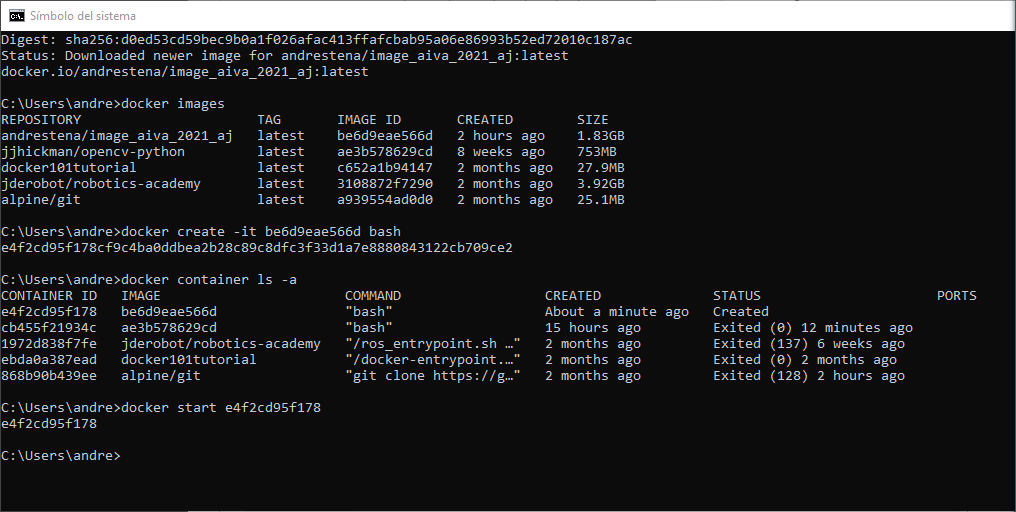
Paso 2: Ejecutar el comando *docker pull andrestena/image\_aiva\_2021\_aj:latest* en la cmd y esperar a que se descargue la imagen.

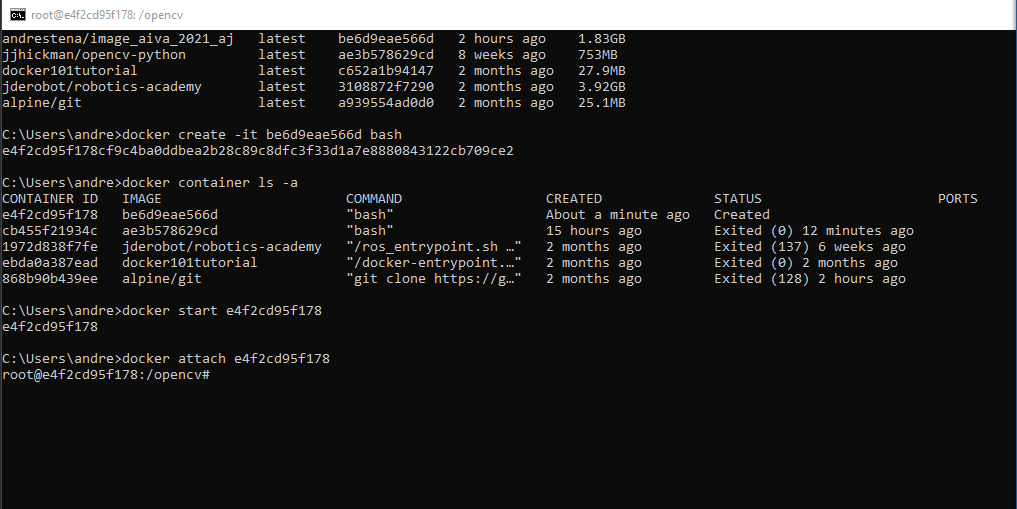
Paso 3: Ejecutar el comando *docker images* en la cmd, una vez ejecutado este comando se deberá fijar en la IMAGE ID del repositorio andrestena/image\_aiva\_2021\_aj.

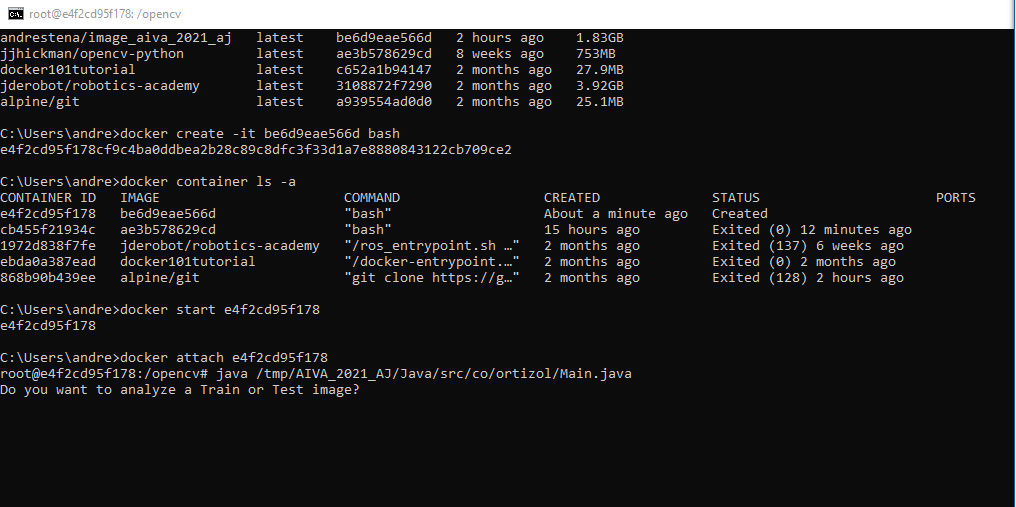
Paso 4: Ejecutar el comando *docker create -it ID bash*, en ID se tendrá que poner el IMAGE ID obtenido en el paso anterior.

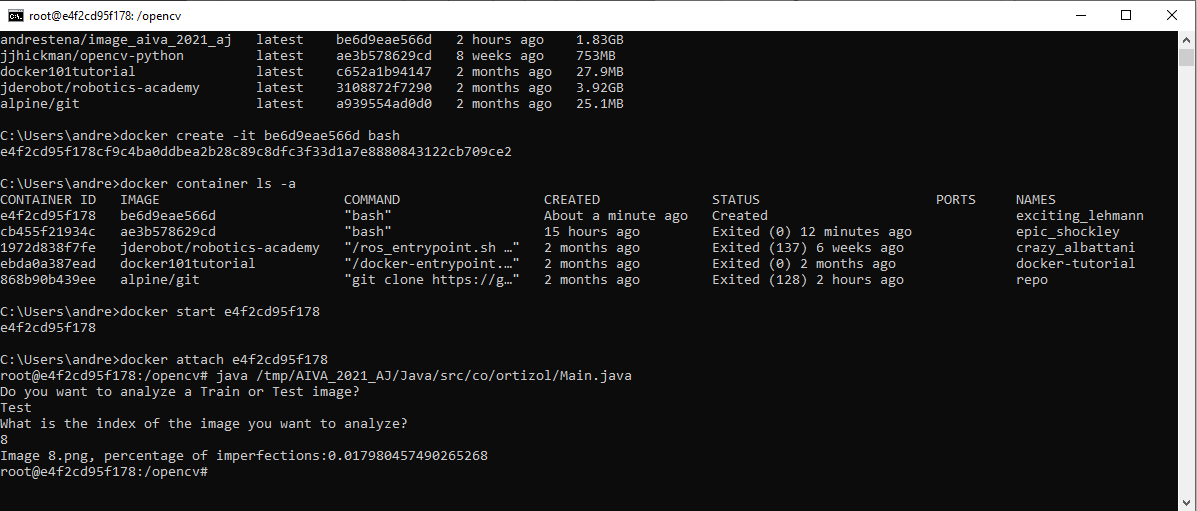


Paso 5: Comprobar que el contenedor ha sido creado, para ello ejecutar el comando *docker container ls -a*.

**Paso 6: A continuación, se utilizará el CONTAINER ID señalado para iniciar nuestro contenedor. Para ello ejecutar *docker start CONTAINER ID.*

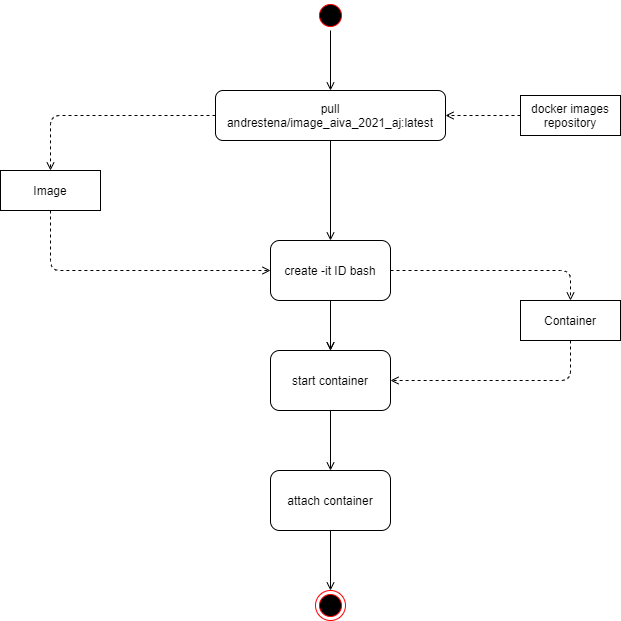
Paso 7: A continuación, se deberá vincular al contenedor ejecutado. Para ello ejecutar *docker attach CONTAINER ID*.

Paso 8: Una vez dentro del contenedor, para ejecutar la aplicación será necesario ejecutar el siguiente comando *java /tmp/AIVA\_2021\_AJ/Java/src/co/ortizol/Main.java.*

Paso 9: Finalmente el algoritmo nos preguntará donde se encuentra la imagen que quieres analizar si en Train o en Test, además del ID de esta.

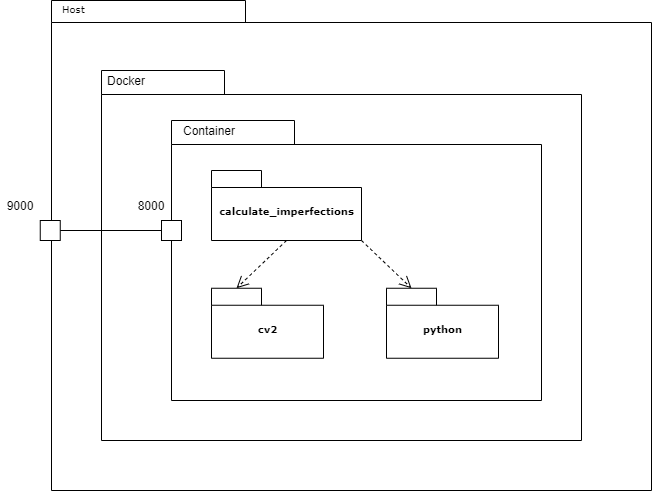
1. Despliegue

En esta sección se van a mostrar tanto el diagrama de despliegue de la aplicación en Docker y para ello será necesario mostrar como es el diagrama de despliegue de Docker.



*Figura 1: Despliegue docker.*

Visto como es el diagrama de despliegue de Docker, a continuación se pasará a mostrar el diagrama de la aplicación dentro de un contendor Docker.



*Figura 2: Despliegue aplicación.*

Una vez se haya creado el contenedor de Docker que dispone de la aplicación, se podrá ejecutar la aplicación sin ningún tipo de problema ya que el contenedor también posee las dependencias que requiere el método *calculate\_imperfections*.

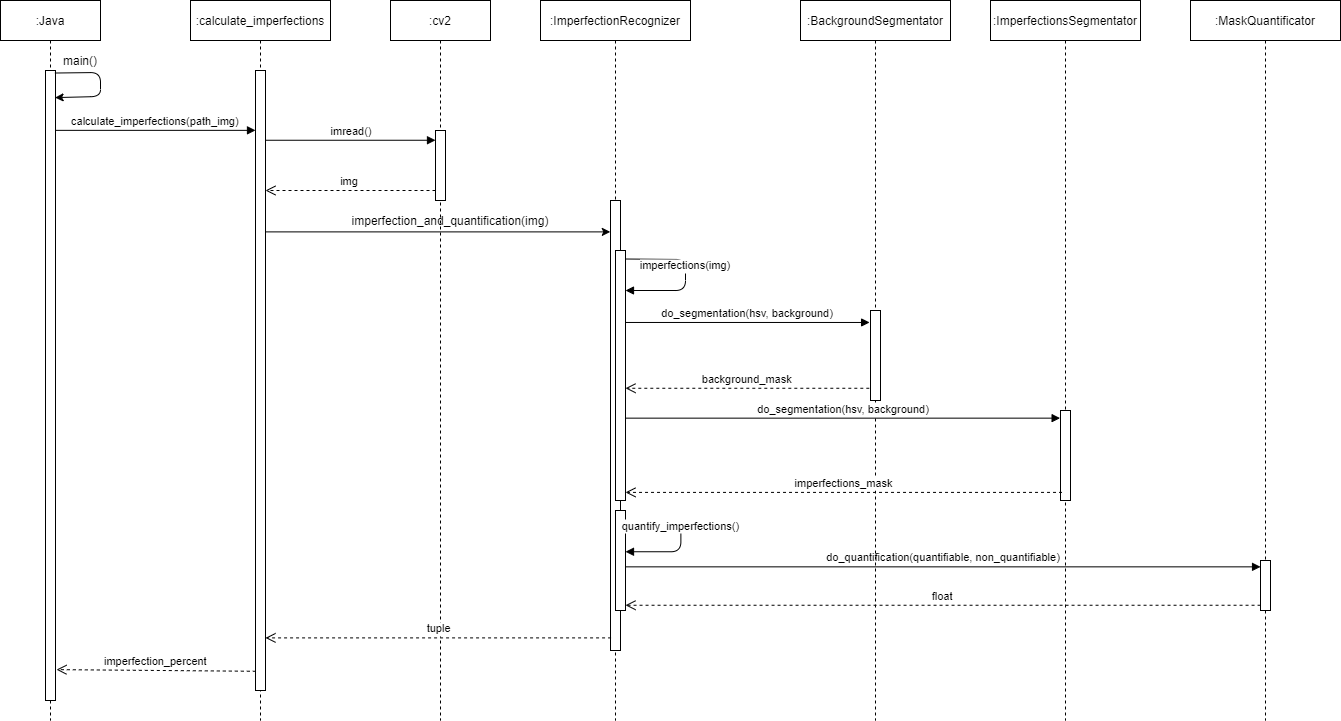
1. Funcionamiento del sistema

La aplicación sigue el siguiente esquema, en primer lugar, se ejecuta un archivo Java el cuál pedirá la ruta de la imagen a analizar. Esta ruta será enviad al método *calculate\_imperfections* que leerá la imagen proporcionada utilizando la librería de opencv. A continuación, llamará al método *imperfection\_and\_segmentation(img)* de la clase ImperfectionRecognizer que devolverá una tupla con la máscara y el porcentaje de área con imperfecciones.

Para calcular dicha máscara ImperfectionRecognizer llama al método *do\_segmentation(hsv, background)* de BackgrounSegmentator que devolverá una máscara que representa el fondo de la imagen. De tal manera que discriminaremos así todo lo que no sea la pieza de madera.

Dicha máscara del fondo será necesaria para calcular la máscara de imperfecciones. Para ello ImperfectionRecognizer activa el método *do\_segmentation(hsv, background)* de la clase ImperfectionSegmentator.

Una vez hemos calculado la máscara de imperfecciones, es necesario cuantificar que área abarcan dichas imperfecciones. Para esto ImperfectionRecognizer utilizará el método *do\_quantification(quantifiable, non\_quantifiable)* de la clase MaskQuantificator.

Finalmente, el porcentaje de imperfecciones de la imagen calculado será devuelto a Java para que este lo muestre por pantalla.

*Figura 3: Diagrama de secuencia de la aplicación.*