21-4-2020

FALLOS EN SUPERFICIES METÁLICAS

Documento 3: SISTEMA FUNCIONAL

PATRICIA FERNÁNDEZ FERNÁNDEZ
GARAZI ALFARO GARCÍA

Contenido

1. INTRODUCCIÓN

A lo largo de este documento, se mostrará una etapa de desarrollo *software* que juega un papel fundamental en la evolución del cualquier proyecto informático.

En esta sección de implementación y despliegue, se mencionarán temas como la comunicación cliente/servidor, la interacción del cliente con el sistema, técnicas que se emplean para entregar el producto como este exige, así como técnicas que se utilizan para satisfacer los objetivos y el producto de este.

El sistema en general se responsabiliza de detectar 6 tipos de fallos en superficies de piezas metálicas, a partir de imágenes tomadas en la línea de producción. Como inteligencia del sistema, se emplea una red neuronal de detección YOLO, la cual devolvería el producto final, un archivo con formato *.xml* que contiene la clase de imperfección de las piezas defectuosas y su localización en la imagen.

Para ello, se hará uso de aplicaciones de cliente/servidor, la ejecución del software del sistema se implementará en un entorno servidor, a fin de adquirir la mayor comodidad y simplicidad por parte del cliente.

1.1. DESCRIPCIÓN DE LA INSTALACIÓN

En cuanto a la instalación del sistema en la empresa, se realizará de una forma simple y efectiva.

Por un lado, la empresa dispone de la instalación de una cámara orientada a capturar cada pieza metálica que transita por la línea de producción. Además, dichas imágenes tomadas se almacenan en un directorio del PC ubicado en un punto cercano del proceso de verificación de defectos. Por lo tanto, no sería necesario la disposición ni montaje de ningún componente hardware adicional.

Por otro lado, se realizaría la configuración de un programa en el PC del cliente, realiza una petición al servidor de analizar cada imagen almacenada del directorio con técnicas de visión artificial. Como respuesta, recibiría un archivo XML asignada a cada pieza.

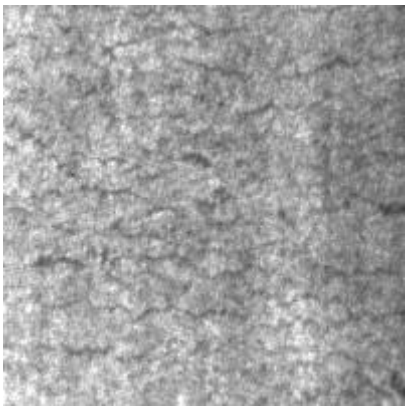

Figura 2: Imagen de una pieza metálica (input).

```
<annotation>
        <folder>cr</folder>
        <filename>crazing_1.jpg</filename>
        <source>
                <database>NEU-DET</database>
        </source>
        <size>
                <width>200</width>
                <height>200</height>
                <depth>1</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>crazing</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>2</xmin>
                        <ymin>2</ymin>
                        <xmax>193</xmax>
                        <ymax>194</ymax>
                </bndbox>
        </object>
</annotation>
```

Figura 2: Fichero XML (output).

La instalación y configuración de todo el sistema seguiría los siguientes pasos:

1. **INSTALACIÓN DE LIBRERIAS EN EL SERVIDOR:** Se debe de instalar la versión Python 3 y una serie de librerías que aparecen en el documento de requirements.txt, se obtendrían lanzando el comando *pip3 install requeriments.txt.* Además, se debe de llevar a cabo una instalación manual de CUVA (NVIDIA) en una versión igual o mayor a la 10.0.
2. **INTRUCCIONES DE EJECUCIÓN DEL SERVIDOR:** Una vez dentro del repositorio de *AIVA_MUVA_DefectosMetales,* se ejecuta el comando *python3 main.py*.
3. **INSTALACIÓN DE LIBRERIAS DEL CLIENTE:** La instalación en el entorno del cliente se va a realizar de la misma manera que en el servidor, sin embargo, no haría falta la instalación de CUDA (NVIDIA). Es decir, se instala Python 3 y se ejecuta el comando *pip3 install requeriments.txt.*
4. **INSTRUCCIONES DE EJECUCIÓN DEL CLIENTE:** Dentro de la carpeta *AIVA_MUVA_DefectosMetales,* el software del cliente se ejecutará con el comando *python3 client.py*.

En definitiva, la instalación y configuración del programa en el entorno del cliente será breve, dado que el *sofware* más pesado y computacionalmente más pesado se efectuará en el servidor.

1.2. ENLACE AL PROYECTO DE GITHUB

El siguiente enlace conduce al ecosistema del proyecto de Github: https://github.com/pfernandezferna1993/AIVA_MUVA_DefectosMetales/issues

2. DESPLIEGUE

En este apartado de despliegue, se va a mostrar la distribución física de los componentes *software* en los diferentes elementos físicos (nodos) que forman el sistema, a fin de ver de qué modo se sitúan estos componentes lógicos en los distintos nodos.

Gracias al diagrama de despliegue se visualizará dicha distribución de una manera más representativa, además de ofrecer una visión general de como está desplegado el sistema de información.

2.1. DIAGRAMA UML DE DESPLIEGUE

El siguiente diagrama, mostrado en el *Diagrama 1,* expone cómo funciona la aplicación cliente-servidor que soluciona el problema de detectar fallos en superficies metálicas.

El programa cliente carga una imagen de la base de datos, que transforma a formato JSON, después de haberla codificado en base64. Este archivo JSON, formará los datos que se enviarán mediante una petición POST al servidor.

El servidor recibe dicho POST, y vuelve a decodificar la imagen contenida en el JSON recibido. Posteriormente, se el servidor utiliza el detector de defectos en piezas metálicas, introduciendo cómo argumento la imagen recibida del cliente. La salida del detector será un archivo *.xml,* cuyo contenido está en formato *Pascal VOC* e incluye el resultado de la detección. Dicho fichero irá contenido en el mensaje, en respuesta al POST, que el servidor manda al cliente.

El cliente analizará el contenido de dicha respuesta y la mostrará en pantalla.

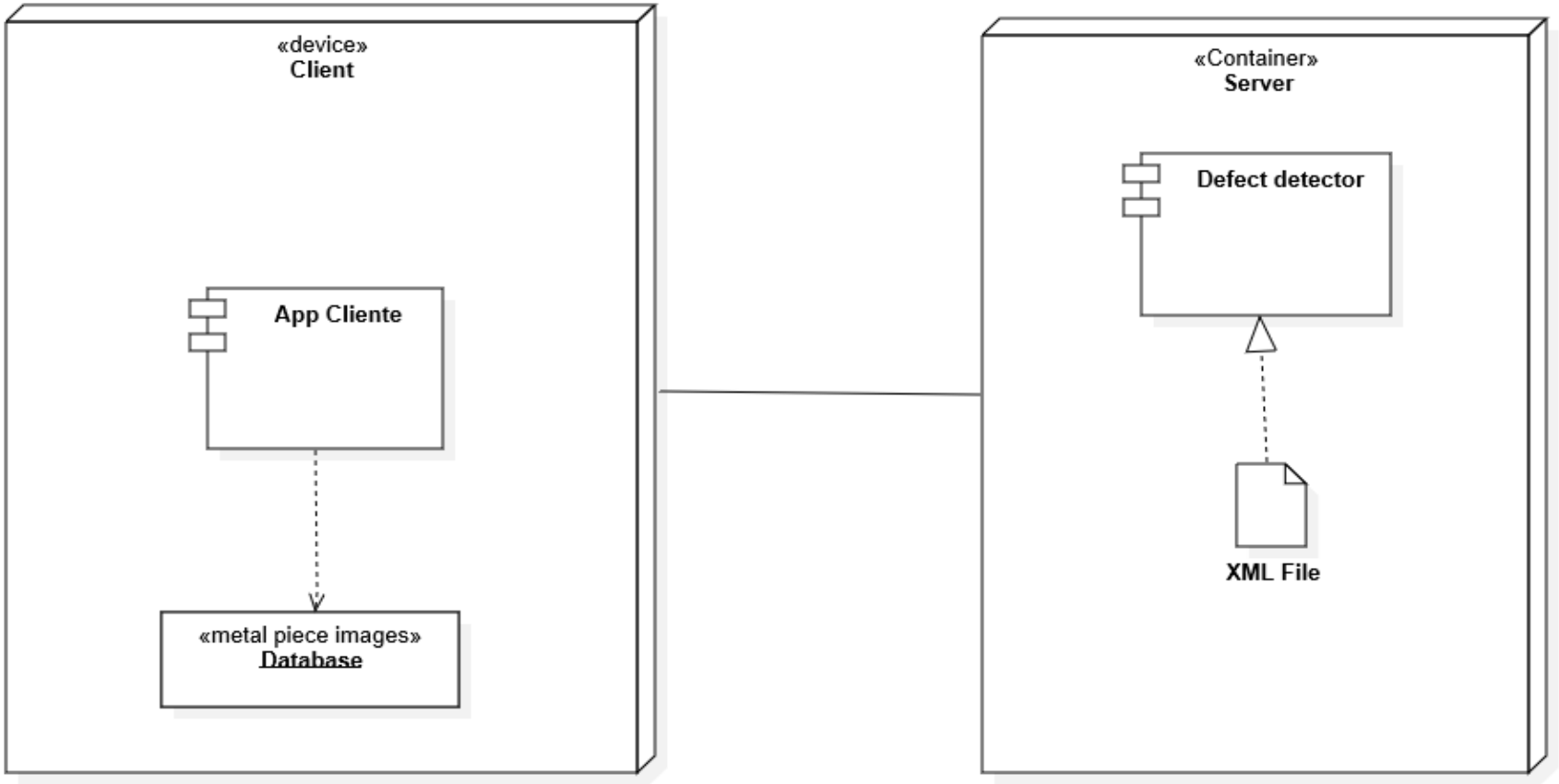

Diagrama 1: Diagrama UML de Implementación.

3. FUNCIONAMIENTO DEL SISTEMA

Se describirá el funcionamiento del sistema a través de un diagrama UML de secuencia, con el objetivo de interpretarlo de una manera más visual y representativa.

3.1. DIAGRAMA UML DE SECUENCIA

En el *Diagrama 2*, podemos ver el diagrama de secuencia final del desarrollo requerido para este proyecto. Este diagrama, incluye ya la diferenciación entre el Cliente ("App") y el Servidor, que son los desarrollos que se comunicarán entre ellos para realizar el procesado requerido.

- **Paso 1:** Dentro del Servidor, en el desarrollo interno del procesado de imágenes generado para detectar defectos en piezas metálicas, se cargará el modelo de red neuronal (Yolo V3), en el cual se cargan los nuevos pesos actualizados.
- **Paso 2:** El Cliente carga una imagen, que codifica y transforma a formato JSON.
- **Paso 3:** El Cliente envía una petición POST al "Controlador" (Servidor http) en el que incluye el JSON generado en el paso anterior.
- **Paso 4:** El "Controlador" (Servidor http), que recibe el POST, descodifica los datos y extrae la imagen para enviársela al detector de defectos.
- **Paso 5:** El detector recibe la imagen y se la envía al modelo de red neuronal Yolo V3, para realice la clasificación de sus defectos y la regresión que localiza estos en la imagen.
- **Paso 6:** La red, envía al detector, los resultados obtenidos en la clasificación y regresión requeridas, en formato Pascal VOC dentro de un archivo *.xml*.
- **Paso 7:** El detector recibe el resultado enviado por la red y se lo transmite al "Controlador" (Servidor http).
- **Paso 8:** Por último, el "Controlador" (Servidor http) transforma la respuesta obtenida del detector a formato json, enviándola cómo respuesta al POST recibido del Cliente.

En este diagrama queda así ilustrado de manera clara, todo el proceso de intercambio de datos entre los diferentes softwares desarrollados para este proyecto.

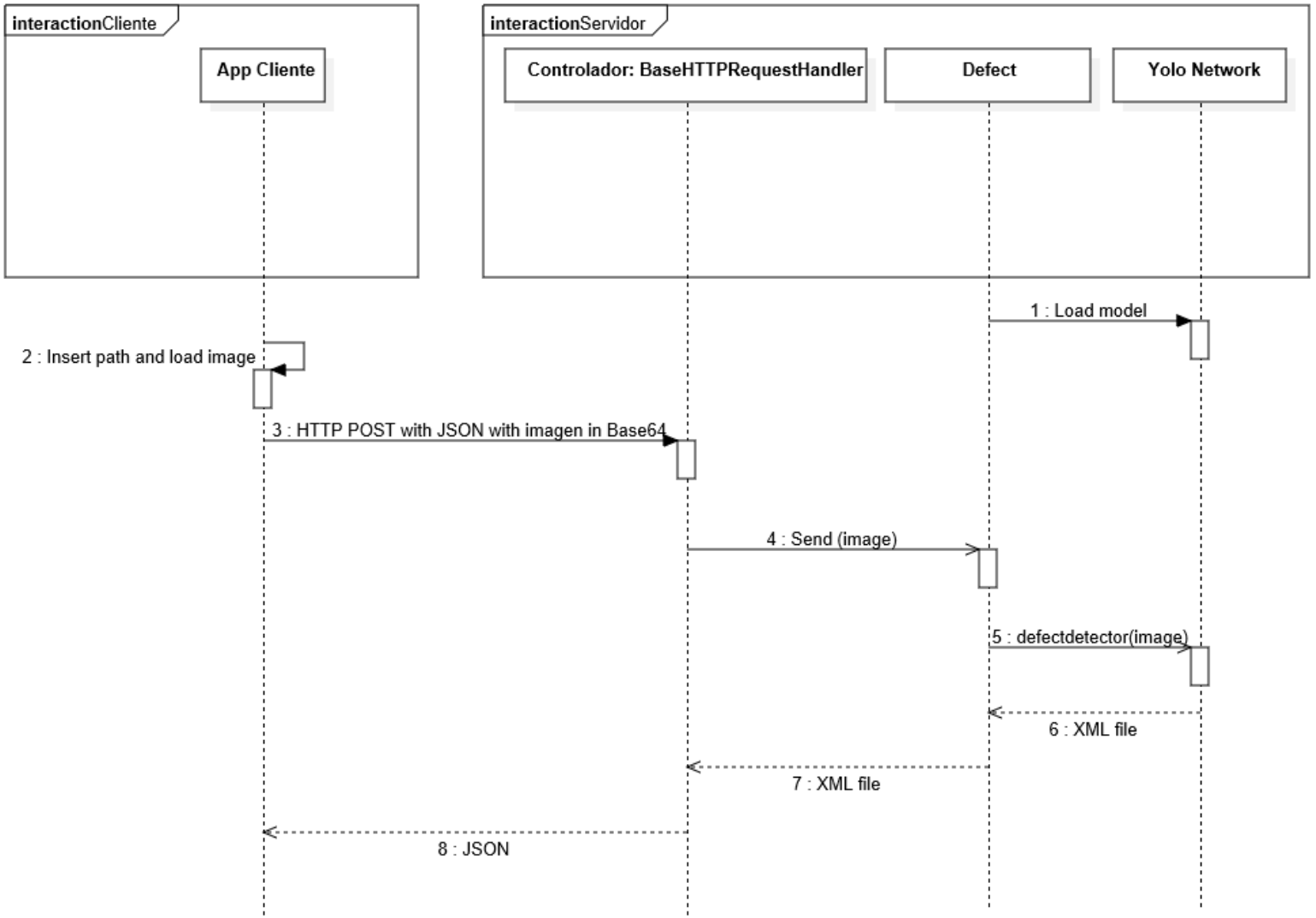

Diagrama 2: Diagrama UML de secuencia.