AIVA-Reconocimiento de crotales.

Entregable 3

2020

ABRIL 20

MUVA
Creado por:
Roberto Gallardo Cava
César González Fernández

Universidad Rey Juan Carlos

Contents

Introducción

En esta fase del proyecto, la aplicación desarrollada será capaz de extraer el identificador del animal a partir de una imagen del crotal de dicho animal, a partir de una aplicación web. Esta aplicación web contará con una interfaz intuitiva (**Figura 1**) en la que el usuario podrá subir una imagen que será procesada por el servidor, y recibirá como respuesta el identificador que aparece en dicha imagen (**Figura 2**).

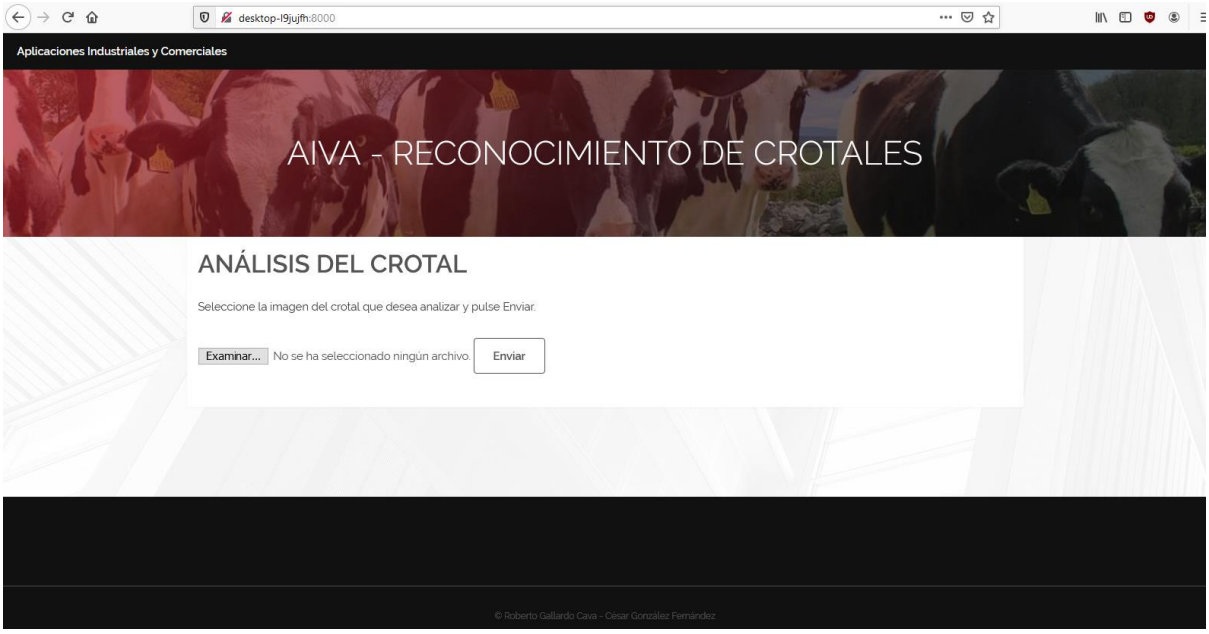

Figura 1. Pantalla inicial de la aplicación web.

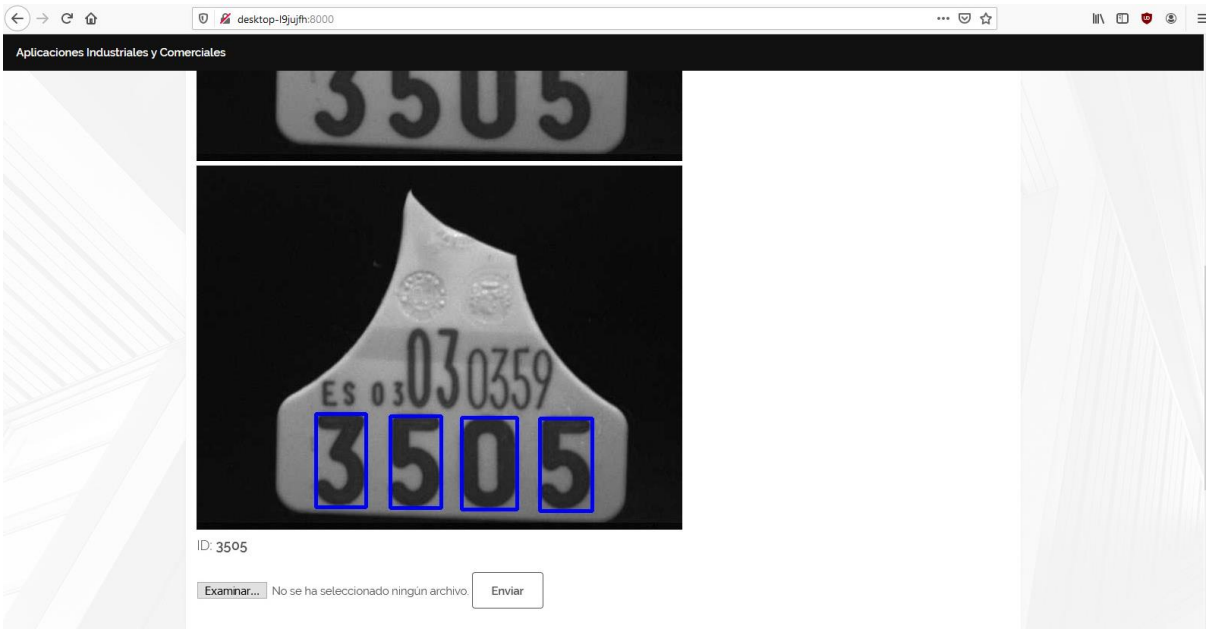

Figura 2. Pantalla con el resultado devuelto por el servidor tras realizar una identificación.

La aplicación seguirá un diseño cliente-servidor en el que la aplicación web hará de cliente, mientras que el servidor ejecutará la aplicación desarrollada para identificar crotales.

La aplicación web ha sido desarrollada utilizando tecnologías web como *HTML y JavaScript.* Por su parte, el lado del servidor ha sido desarrollado utilizando el framework de Python *Flask* y el servidor *wsgi Waitress.* Además de esto, y para facilitar el despliegue de dicha aplicación, se ha creado una imagen de *Docker (*plataforma de contenedores dominante en el mercado) a partir de la cual se podrá ejecutar un contenedor que contendrá la aplicación.

Dado que los crotales que se pretenden identificar no siguen ningún estándar, se ha decidido asumir que el identificador de un crotal es aquel número que aparece en la zona baja del mismo, y que, por regla general, es además el de mayor tamaño.

Esta aplicación puede ser descargada desde el repositorio de *GitHub* del proyecto: https://github.com/RoberG/AIVA-Reconocimiento-de-crotales

Instalación

La aplicación esta lista para ser ejecutada de dos maneras diferentes según sean las necesidades. La primera (recomendada) permite la ejecución de la aplicación dentro de un contenedor de *Docker* mientras que la segunda ellas es ejecutar el servidor *waitress* directamente sobre el sistema operativo.

Docker (recomendada)

Para facilitar el uso y despliegue de la aplicación, se ha creado una imagen *Docker* almacenada en el repositorio oficial *DockerHub.* Esto permite tener de manera rápida un servidor ejecutando la aplicación web sin necesidad de instalar las dependencias del proyecto en la máquina.
En primer lugar, es necesario tener instalada la aplicación *Docker* en la maquina que hará de servidor. Para ello, hay que seguir las instrucciones que se indican en la página oficial https://docs.docker.com/get-docker/ seleccionando el sistema operativo apropiado.
Una vez nos aseguremos que *Docker* esta instalado y funciona correctamente, únicamente habrá que crear un contenedor que use la imagen de la aplicación

disponible en los repositorios oficiales de *Docker.* Si trabajamos en un entorno Linux, esto se hará ejecutando la siguiente el siguiente comando:

```
docker run -it -p 8000:8000 sario/aiva-reconocimiento-crotales
```

Esto creará y ejecutará un nuevo contenedor Docker que ejecuta la aplicación, que será accesible a través del puerto 8000 de la máquina que ejecuta el servidor.

Servidor waitress

La segunda opción disponible para ejecutar la aplicación es levantar el servidor directamente sobre el sistema operativo de la máquina. Para ello, primero habrá que instalar la aplicación como se indica en la sección correspondiente en el fichero README. Una vez instalado, para levantar el servidor ejecutamos el siguiente comando:

```
python -m reconocimiento_crotales.server
```

Tras terminar de iniciarse el proceso, la aplicación estará disponible a través del puerto 8000 de la propia máquina.

Despliegue

En la **Figura 3** se puede apreciar el diagrama UML de Despliegue que refleja el diseño de la arquitectura web desarrollado para la aplicación.

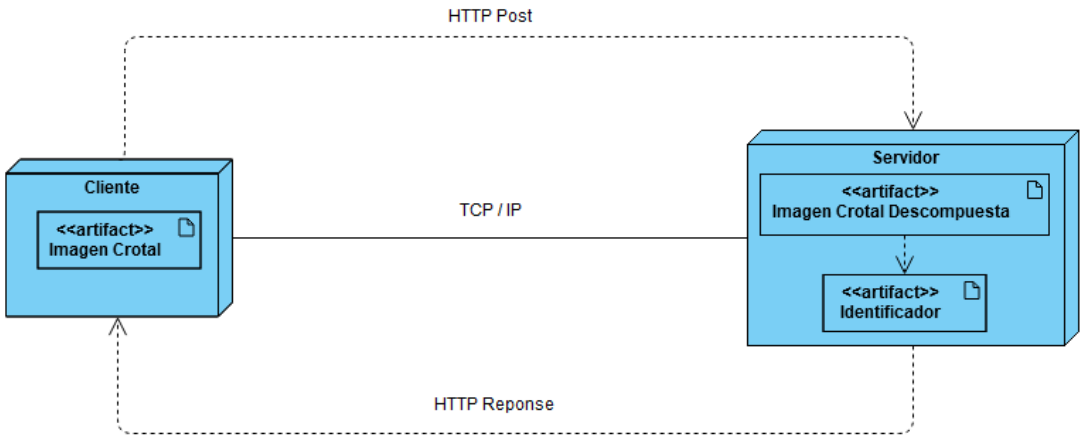

Figura 3. Diagrama de Despliegue

El cliente mediante una petición HTTP Post, le enviará al servidor la imagen del crotal que la que se desee reconocer el identificador. El Servidor recibirá la imagen, y utilizando la algoritmia previamente explicada en anteriores entregas, acabará detectando las distintas bounding boxes (Bbox) del identificador en la imagen. El servidor devolverá la imagen con las Bbox detectadas y el identificador, que serán mostrados en el cliente.

Funcionamiento del sistema

La **Figura 4** presenta un diagrama de secuencia que muestra el comportamiento del sistema. En dicho diagrama se muestra la interacción entre el cliente y el servidor.

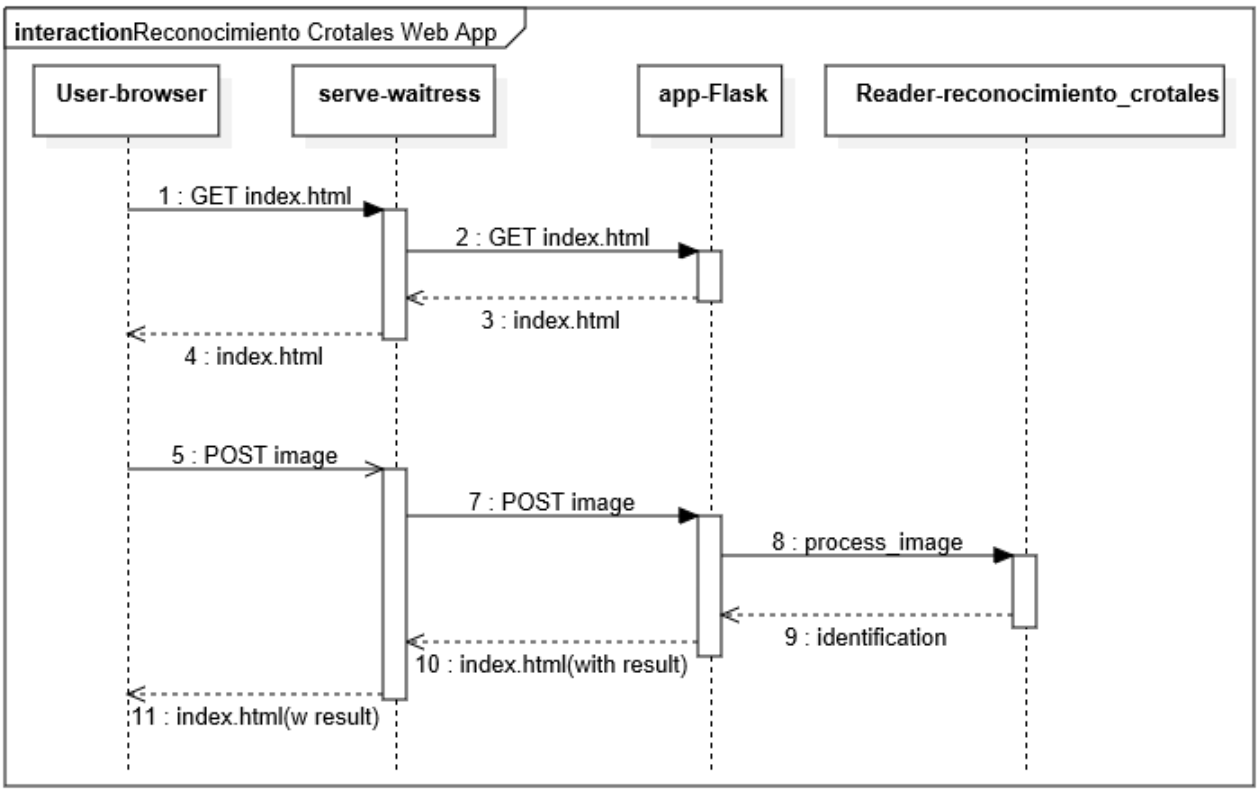

Figura 4. Diagrama de la aplicación web.

El usuario puede acceder a la aplicación web desde cualquier navegador. El navegador, realiza en primera instancia una petición GET al servidor para que este le devuelva la página de inicio de la aplicación y así poder presentársela al usuario. El servidor *wsgi (waitress)* recibe la petición y se la transimite a la *app,* que es un objeto del paquete de Python *Flask.* Esta *app* responde a la petición anterior devolviendo la página, que, al tratarse de contenido estático, puede gestionar por si misma. La página (**Figura 1**), en formato *HTML* es devuelta al navegador del usuario a través del servidor de *waitress.*
Una vez la pagina es presentada al usuario, este puede cargar una imagen para que el servidor trate de identificarla. Esta imagen se envía a través de una petición *POST* de *HTTP.* La petición, al igual que ocurría con la anterior, llega al servidor *waitress,* el cual se la pasa a la *app* de *Flask*. En esta ocasión, el objeto *app* no es capaz de gestionar la petición por si sola, por lo que se apoya en el paquete *reconocimiento_crotales*

desarrollado en este proyecto. Para ello, utiliza un *Reader* para procesar la imagen. El funcionamiento de este *Reader* es el mismo que el explicado en el segundo entregable. Con la imagen ya identificada, el resultado se devuelve a la *app,* la cual la renderiza en un template *HTML* para que pueda ser presentado al usuario. Esta nueva página (**Figura 2**) es enviada desde la *app* de *Flask* hasta el navegador del usuario a través de nuevo del servidor *Waitress.*