Realizado por Álvaro Manuel Aparicio Morales

Tutorizado por Daniel Garrido Márquez y Cristian Martín Fernández

Departamento de Lenguajes y Ciencias de la Computación

MÁLAGA, febrero de 2021

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DEL SOFTWARE

ANÁLISIS DE IMÁGENES EN ARQUITECTURA CLOUD-FOG CON LWM2M *(LIGHT WEIGHT MACHINE TO MACHINE)*

IMAGE ANALYSIS IN CLOUD-FOG ARCHITECTURE WITH LWM2M *(LIGHT WEIGHT MACHINE TO MACHINE)*

Realizado por
Álvaro Manuel Aparicio Morales

Tutorizado por
Daniel Garrido Márquez
Cristian Martín Fernández

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, FEBRERO DE 2021

Agradecimientos

Quiero agradecer a los profesores que he tenido por haber fomentado el desarrollo de mi curiosidad por la Informática, y quiero hacer especial mención a Daniel Garrido Márquez y a Cristian Martín Fernández por haberme dado la oportunidad de realizar este proyecto y por su tutorización a lo largo de éste.

También agradecer a mi familia, en especial a mis padres y a mi hermano por dedicarme su tiempo y todo su esfuerzo durante mi trayectoria académica.

También me gustaría agradecer a mis amigos por estar ahí y apoyarnos entre nosotros.

Resumen

Las limitaciones de los dispositivos de Internet of Things (IoT), fueron uno de los diversos motivos que impulsaron el desarrollo de las infraestructuras de Computación en la Nube para dar soporte a las aplicaciones de uso intensivo de datos de una forma escalable. Sin embargo, los problemas de tiempo como la latencia no son considerados en estos sistemas. En esta cuestión es dónde entra en juego la arquitectura Fog Computing, reduciendo la cantidad de datos transmitidos a la nube y balanceando las necesidades entre los dos sistemas, así como disminuyendo el tiempo de latencia y respuesta a los eventos. Esta problemática trajo la necesidad de desarrollar una arquitectura cloud-fog-edge que permita a aplicaciones críticas garantizar la tolerancia a fallos.

En este trabajo de fin de grado se han desarrollado los componentes para evaluar el rendimiento de la plataforma tolerante a fallos del grupo de investigación ERTIS, bajo una mayor carga de trabajo mediante la realización de análisis de imágenes. Para llevar a cabo dicha finalidad, se ha desarrollado un servicio cuya funcionalidad principal es la transmisión de diversas imágenes y datos sobre éstas (nombre y fecha de emisión) mediante el protocolo de comunicación Light Weight Machine to Machine (LWM2M). Estas imágenes y sus respectivos datos, se solicitan a través de una aplicación para realizar un análisis que consiste en la detección de los objetos que se pueden observar en la imagen recibida. El resultado de este análisis se podrá comprobar mediante una página web que mostrará dicho resultado.

Palabras clave:

Internet de las Cosas, Light Weight Machine to Machine, Análisis de Imágenes, Fog Computing, Docker

Abstract

The limitations of the Internet of Things (IoT) devices were one of the various reasons that drove the development of Cloud Computing infrastructures to support data-intensive applications in a scalable way. However, time issues such as latency, are not considered in these systems. This is where the Fog Computing architecture comes into play, reducing the amount of data transmitted to the cloud and balancing the needs between the two systems, as well as reducing the latency and response time to events. This problem led to the need to develop a cloud-fog-edge architecture that allows critical applications to guarantee fault tolerance.

In this final degree project, the components to evaluate the performance of the platform already developed under a greater workload by performing image analysis have been implemented. To carry out this purpose, a virtual embedded system has been developed, which main functionality is the transmission of several images and their data (name and date of issue) through the *Light Weight Machine to Machine (LWM2M)* communication protocol. These images and their respective data are requested through an application to perform an analysis that consists of detecting the objects that can be observed in the received image. The result of this analysis can be verified through a web page that will show the result.

Keywords:

Internet of Things, Light Weight Machine to Machine, Image Analysis, Fog Computing, Docker

Índice

Índice de Figuras

1

Introducción

En la actualidad, el Internet de las Cosas (*Internet of Things, IoT*) está cada vez más presente, desde los sistemas de monitorización de aplicación industrial, hasta electrodomésticos que a una orden de nuestro dispositivo móvil pueden prepararnos un café. Pero para poder llegar a este momento, se ha tenido que ir produciendo una serie de avances en el campo tanto de la informática como en el de las telecomunicaciones, comenzando con la creación de Internet hasta el que se considera el primer dispositivo conectado, una tostadora en el año 1990.

Pero esta idea no es algo de pocas décadas, en realidad se remonta hacia el año 1874 donde un grupo de científicos franceses, para llevar a cabo experimentos sobre telemetría, instalaron dispositivos de meteorología y de profundidad de nieve en el Mont Blanc. Aunque concepto de dispositivos conectados, se puede ver posteriormente plasmada en pensamientos y escritos de Nikola Tesla o Alan Turing.

"Cuando lo inalámbrico esté perfectamente desarrollado, el planeta entero se convertirá en un gran cerebro, que de hecho ya lo es, con todas las cosas siendo partículas de un todo real y rítmico… y los instrumentos que usaremos para ellos serán increíblemente sencillos comparados con nuestros teléfonos actuales. Un hombre podrá llevar uno en su bolsillo" Nikola Tesla, 1926.

"…también se puede sostener que es mejor proporcionar la máquina con los mejores órganos sensores que el dinero pueda comprar, y después enseñarla a entender y hablar inglés. Este proceso seguirá el proceso normal de aprendizaje de un niño" Alan Turing, 1950.

1.1 Motivación

La principal misión del IoT, es la unión entre el mundo físico y el mundo digital gracias a la utilización de dispositivos que nos permite cuantificar el entorno que nos rodea y producir una respuesta en función de la información que se recopila a través de estos. Hoy en día, estos dispositivos, tienen un gran impacto en sectores como el agrícola, sanitario, energético y por su puesto en la nueva Industria 4.0. Algunas de las aplicaciones de estos dispositivos son la detección de incendios forestales, sistemas para la detección de inundaciones, seguimiento de ganados, mantenimientos predictivos de maquinaria…, estas aplicaciones requieren de una continua generación de datos que se recopilan de los distintos sensores que tienen los dispositivos y uno de los problemas que acarrea la utilización de estos sistemas es la limitación de la capacidad de cálculo de estos dispositivos. Este junto con otros motivos impulsaron el desarrollo de la Computación en la Nube. Sin embargo, los tiempos de latencia pueden interferir en la respuesta otorgada por una aplicación crítica que hace uso de estos sistemas. Para poder solventar dicho problema, se requiere de la utilización de una arquitectura *Fog Computing,* la cual permite reducir los tiempos de respuestas debido a la proximidad de estas infraestructuras a los sistemas IoT. Esta problemática provocó la necesidad de desarrollar una arquitectura *cloud-fog-edge* que proporcione a las aplicaciones críticas un entorno tolerante a fallos tanto a nivel hardware como a nivel software.

Este trabajo se centra en la evaluación de la arquitectura *cloud-fog-edge* del grupo de investigación ERTIS tolerante a fallos hardware y software para comprobar la robustez y fiabilidad de este sistema cuando soporta una gran cantidad de trasmisión de datos como son las imágenes.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado es evaluar el comportamiento de la arquitectura *cloud-fog* tolerante a fallos software y hardware sometiéndola a una mayor carga de trabajo a través de análisis de imágenes. Para poder lograr este objetivo se han establecido cinco subobjetivos.

El primer subobjetivo, consistirá en transmitir, utilizando el protocolo de comunicación LWM2M, una imagen para su posterior análisis. Para lograr este subobjetivo, se debe hacer uso de la librería de programación Leshan, que implementa

el protocolo de comunicación mencionado anteriormente mediante el lenguaje de programación Java.

El segundo subobjetivo, es realizar un análisis de la imagen, mediante un algoritmo de detección de objetos. Para poder llevar a cabo este subobjetivo, debemos desarrollar un servicio que sea capaz de establecer una comunicación entre el primer módulo (resultante del primer objetivo) y el módulo resultante de este subobjetivo, a través de la arquitectura tolerante a fallos.

El tercer subobjetivo es proveer de un sistema de almacenamiento para guardar el resultado del análisis de imágenes y posteriormente poder visualizarlo en una aplicación web.

El cuarto subobjetivo, se trata de desarrollar una aplicación web que se encarga de mostrar, tras un proceso de identificación de usuario, las imágenes analizadas que se encuentran en el sistema de almacenamiento.

El último subobjetivo es desplegar el producto resultante, junto con la arquitectura, haciendo uso de la tecnología de contenerización Docker.

1.3 Estructura de la memoria

La memoria se estructura en cinco capítulos aparte de la introducción en los que se describe todo el proceso llevado a cabo para la elaboración del proyecto. Por último se incluyen las referencias y los tres apéndices finales de *Manual de Instalación, Manual de Uso* y *Notas.*

A continuación, se describe la estructura del documento.

- **Revisión de trabajos y proyectos similares:** Se procederá al análisis de diversos trabajos y a la mención de las diferencias entre estos y el trabajo presente.

- **Tecnologías y herramientas utilizadas:** En este capítulo se presentan el entorno tecnológico llevado a cabo para la elaboración del proyecto.

- **Etapas del desarrollo del trabajo:** En este capítulo se recoge la explicación sobre las fases llevadas a cabo para el análisis, especificación, diseño e implementación del proyecto.
- **Conclusiones:** En este capítulo se recogen las conclusiones, la experiencia personal durante la elaboración y se procede a evaluar las posibles mejoras que se pueden realizar en el presente trabajo.

2

Revisión de trabajos similares

Análisis de Imágenes

Hoy en día, es frecuente encontrar sistemas y aplicaciones que hagan uso de este tipo de análisis. Cada vez existen más herramientas que necesitan procesar las imágenes para llevar a cabo su principal funcionalidad. En definitiva cualquier sistema que haga uso de inteligencia artificial para este procesamiento es susceptible de asemejarse a nuestro proyecto.

Posibles casos de uso de este tipo de análisis los podemos observar en los coches autónomos, pues necesitan realizar un rápido análisis de las imágenes en tiempo real de las cámaras para poder efectuar una determinada acción. Esto es posible llevarlo a cabo gracias a las arquitecturas de edge computing, pues permiten llevar a cabo este análisis en el propio vehículo.

Otra aplicabilidad puede ser en la Industria 4.0, haciendo uso de cámaras que identifiquen los artículos producidos y puedan identificar si existe o no algún error en la fabricación.

También se puede aplicar esta arquitectura y esta aplicación de análisis de imágenes para modelar y agilizar el tráfico, tanto terrestre como portuario. A continuación vamos a exponer varios proyectos que hacen uso del edge computing y del análisis de imágenes.

REVAMP2T

Es un proyecto que proporciona un seguimiento de peatones a través de análisis de video en una arquitectura Edge.

Esta solución fue presentada en 2019 por Christopher Neff, Matías Mendieta, Shrey Mohan, Mohammedreza Baharani, Samuel Rogers y Hamed Tabkhi.

Para llevar a cabo el seguimiento del peatón hacen uso de dispositivos IoT que contienen cámaras equipadas con Nvidia AGX Xavier que permite realizar el análisis de vídeo mediante deep-learning. Estos dispositivos, agrupados por área, hacen uso de servidores edge para el intercambio de comunicación, procesamiento y almacenamiento.

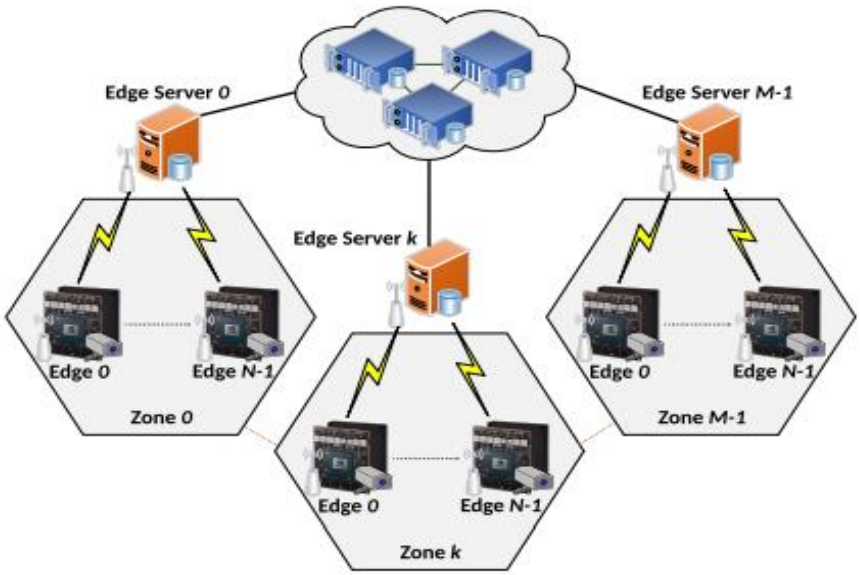

Figura 2. 1: Sistema IoT REVAMP2T

Microsoft Rocket

Microsoft Rocket es una plataforma software cuyo objetivo principal es construir un sistema de tiempo real, bajo coste y preciso para el análisis de video en directo.

Este sistema podrá trabajar bajo una jerarquía geo-distribuida de arquitecturas edge y la nube. Esta plataforma permitirá a cualquier persona con una cámara, disfrutar de forma fácil y asequible, del análisis de vídeo.

La arquitectura de Rocket es configurable para ejecutar en una infraestructura distribuida, que puede abarcar hardware de borde especializado (por ejemplo, Azure Stack Edge) y la nube (por ejemplo, Azure Machine Learning and Cognitive Services).

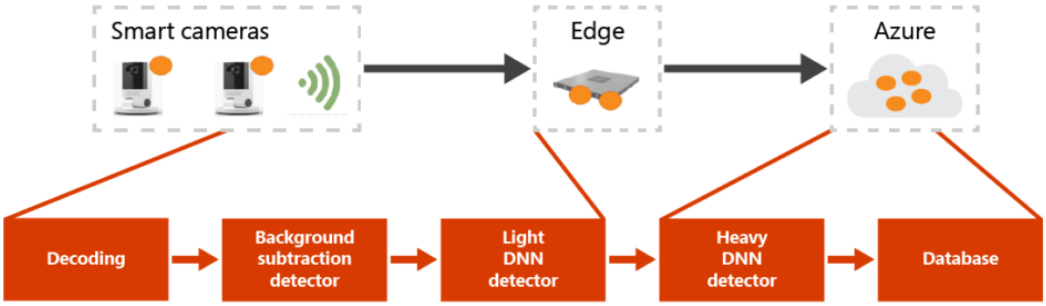

Figura 2. 2: Arquitectura Microsoft Rocket

Critical Care Suite

Para poder agilizar el diagnóstico de neumotórax, Intel y GE-Healthcare han desarrollado un sistema que permite la identificación de neumotórax haciendo uso de algoritmos de inteligencia artificial integrados en el dispositivo para priorizar la revisión crítica de radiografías de tórax.

Para la realización del diagnóstico se hace uso de OpenVINO (Open Visual Inference y Neural Work Optimization). OpenVINO es un kit de desarrollo que facilita la optimización de un modelo de aprendizaje profundo a partir de un marco y la implementación mediante un motor de inferencia en hardware Intel. (Wikipedia Foundation, 2021).

Critical Care Suite escanea automáticamente las imágenes tomadas por el sistema de rayo X. No necesita de ninguna infraestructura adicional ni de redes de comunicación para elaborar el diagnóstico, ya que hace uso de la IA en el propio dispositivo para priorizar la detección de neumotórax.

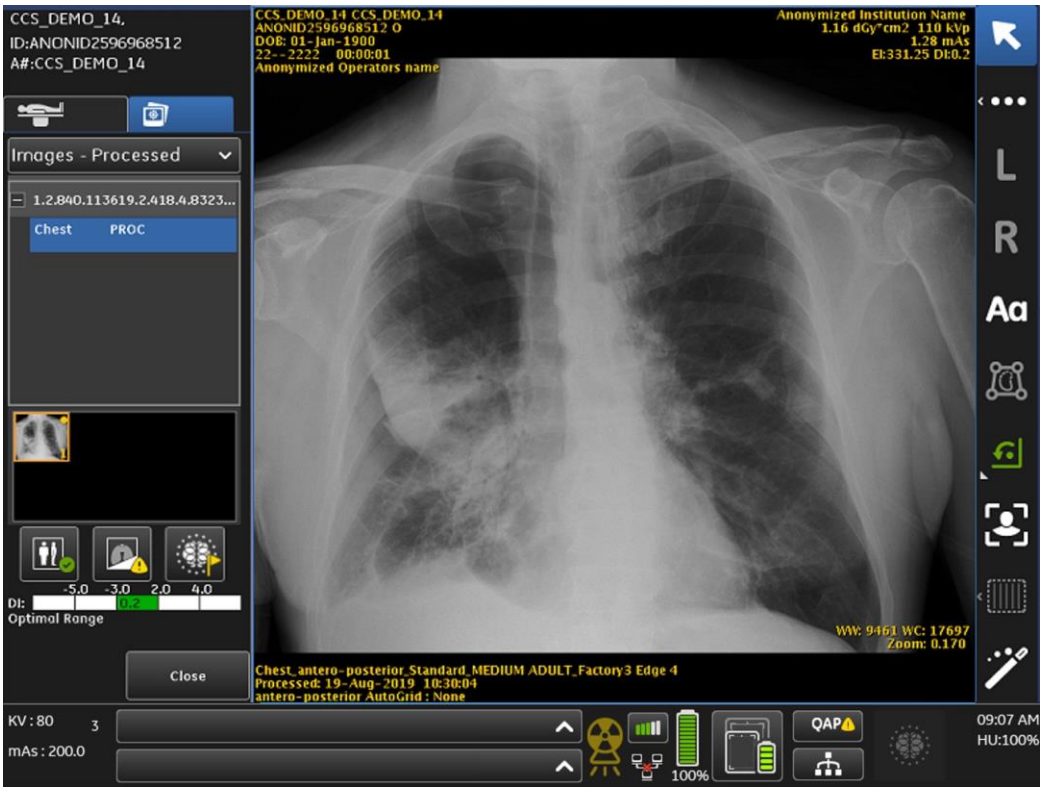

Figura 2. 3: Critical Care Suite

5G Maritime

Es un proyecto que se está llevando a cabo en el Port de Barcelona. Su funcionalidad principal es la de geolocalizar embarcaciones en tiempo real para poder optimizar el tráfico portuario.

Para llevar a cabo dicha finalidad, esta solución hace uso de dos cámaras que transmiten la imagen en tiempo real a un sistema ubicado en la torre de control. La transmisión de la imagen se realiza mediante una red 5G. Este sistema ubicado en la torre de control hace uso de inteligencia artificial para la identificación de los barcos y haciendo uso de un algoritmo consiguen obtener su geolocalización de forma bastante precisa. Este sistema hace uso de la tecnología edge computing.

Como se puede apreciar, este proyecto tiene bastantes puntos en común al que se expone en este trabajo, pues hace uso de unos dispositivos que realizan una captura de imágenes y éstas son transmitidas a una arquitectura, desplegada haciendo uso del edge computing, que contiene una aplicación para la detección de objetos, en este caso detección de barcos.

En los puntos donde difieren, cabe destacar que la transmisión de las imágenes en este proyecto no se realiza mediante una red 5G, ni tenemos un algoritmo para geolocalizar los objetos detectados. Además las imágenes son tomadas en tiempo real, mientras que en nuestro sistema, debemos ordenar la toma de una nueva imagen.

En resumidas cuentas, este proyecto podría ser un caso de aplicación de nuestra arquitectura tolerante a fallos con una aplicación de detección de objetos en imágenes.

Figura 2. 4 : Funcionalidad 5G Maritime

3

Tecnologías y herramientas

En este apartado describiremos las tecnologías y herramientas utilizadas para la elaboración del proyecto.

Light Weight Machine to Machine (LWM2M)

OMA SpecWorks' LigthtweightM2M es un protocolo de comunicación de la Open Mobile Alliance para la comunicación entre máquinas y para el manejo de dispositivos IoT. Este protocolo se construyó en base al standard CoAP[i] y se define en la capa de aplicación proporcionando un estándar comunicativo entre el cliente LWM2M y el servidor LWM2M.

Figura 3. 1: Logo LWM2M

Leshan

Leshan es un proyecto que proporciona librerías, en el lenguaje de programación Java, para el desarrollo de clientes y servidores que hagan uso del protocolo de comunicación *Light Weight Machine to Machine*. Además, Leshan nos proporciona demostraciones de clientes y servidores para conocer mejor el comportamiento de este protocolo.

Figura 3. 2: Logo Leshan

Java

Java es un lenguaje de programación orientado a objetos, que destaca por ser simple, robusto, sólido, con tipado estático y además posee un recolector de basura para liberar memoria de objetos no referenciados. Java también hace referencia a la tecnología que proporciona una máquina virtual que permite ejecutar código compilado Java en cualquier plataforma.

Este lenguaje de programación se ha utilizado para el desarrollo del primer módulo que hace uso de Leshan para desarrollar un cliente que haga uso del protocolo de comunicación *Light Weight Machine to Machine.*

Figura 3. 3: Logo Java

Apache Kafka

Apache Kafka es una plataforma de transmisión de eventos distribuida de código abierto utilizada por miles de empresas para canalizaciones de datos de alto rendimiento, análisis de transmisión, integración de datos y aplicaciones de misión crítica. (Apache Kafka, s.f.)

Algunas de las ventajas destacables de esta plataforma son su alto rendimiento y su escalabilidad.

En el desarrollo de nuestro trabajo, utilizaremos esta plataforma para la transmisión de datos y acciones entre los distintos módulos.

Figura 3. 4: Logo Kafka

OpenCV

OpenCV (*Open Computer Vision*) es una librería *open-source* de visión artificial que proporciona una serie de algoritmos para el tratamiento de imágenes. Esta librería se desarrolló para ser eficiente y usada en aplicaciones de tiempo real. Nuestra aplicación de detección de objetos hace uso de esta librería junto con el sistema de detección de objetos *YOLO* (*You Only Look Once)* para llevar a cabo su finalidad.

Figura 3. 5: Logo OpenCV

Python

Es un lenguaje de programación interpretado multiparadigma, enfocado en la legibilidad del código y es usado tanto para programación orientada a objetos, programación imperativa como para programación funcional. Tiene detrás a una gran comunidad de desarrolladores, que permite la existan de gran cantidad de bibliotecas para implementar aplicaciones de análisis de datos, aplicaciones web, de escritorio… En nuestro trabajo utilizamos la versión de *Python 3.7* .

Figura 3. 6: Logo Python

MongoDB

Es un sistema de gestión de bases de datos NoSQL de código abierto y está orientado a documentos. Este sistema nos ofrece una alta escalabilidad y flexibilidad en los datos.

Se ha utilizado este sistema para el almacenamiento de las imágenes resultantes del análisis, así como para el almacenamiento de los datos de un usuario de la aplicación web. Para la conexión entre la base de datos y las aplicaciones de detección de objetos y web se ha desarrollado una API REST que hace uso de *PyMongo* .[ii]

Figura 3. 7: Logo MongoDB

Flask

Flask es un framework desarrollado en Python. Se utiliza para la creación de aplicaciones web. Entre sus características destaca su facilidad para poder implementar una página web simple, ya que requiere de pocas líneas de código. Este framework se ha utilizado para desarrollar la aplicación web en la que se visualizan las Imágenes analizadas y para la API REST para la conexión con la base de datos.

Figura 3. 8: Logo Flask

Bootstrap

Es un marco de trabajo (framework) de código abierto para el desarrollo *front-end* de las aplicaciones web. Este framework se ha usado para el *front-end* de la aplicación de visualización de imágenes.

Figura 3. 9: Logo Bootstrap

Firebase

Es una tecnología implementada por Google para desarrollar aplicaciones web. (Wikipedia, 2020). Esta plataforma tiene diversas funcionalidades como alojamiento de una base de datos, hosting de páginas web etc. , pero para este proyecto se ha usado su método de autenticación con Google.

Figura 3. 10: Logo Firebase

Eclipse

Es una plataforma de desarrollo diseñada para ser extendida de forma indefinida a través de *plug-ins*, . Nos permite tener distintas perspectivas y vistas, además de poseer integrado un depurador de código (CALENDAMAIA, 2014). Esta plataforma es muy usada en la comunidad de desarrolladores de java, aunque eclipse también admite el desarrollo de aplicaciones en otros lenguajes como Python.

Esta herramienta la utilizamos en nuestro proyecto para el desarrollo del cliente leshan.

Figura 3. 11: Logo Eclipse

Visual Studio Code

VS Code (*Visual Studio Code*) es un editor de código de código abierto (*opensource*), multiplataforma desarrollado por Microsoft con soporte para control de versiones y depuración. Este editor incorpora *plug-ins* que agilizan el proceso de desarrollo además de incorporar herramientas para trabajar con Microsoft Azure y desplegar proyectos. VS Code es el editor que hemos usado para desarrollar tanto la aplicación de detección de imágenes como la aplicación web.

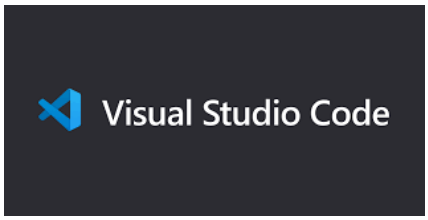

Figura 3. 12: Logo Visual Studio Code

MagicDraw

Es una herramienta *CASE*[iii] compatible con el estándar UML[iv] (*Unified Modeling Language*) que permite realizar modelado de sistemas, software y arquitecturas. Esta herramienta se ha usado para la creación de los diagramas y modelos del cliente Leshan y la aplicación de detección de objetos.

Figura 3. 13: Logo Magicdraw

IFMLEdit

Es un editor web de IFML[v] (*Interaction Flow Modeling Language)* que permite la creación de modelos sobre las interacciones de usuario y el comportamiento de front-end. Este editor se ha usado para el modelo de la aplicación web .

Figura 3. 14: Logo IFML

Docker

Es una tecnología que permite desplegar aplicaciones dentro de contenedores. La contenerización a través de Docker otorga una capa de abstracción entre las aplicaciones y el sistema operativo ya que se encapsula el código de la aplicación y sus respectivas dependencias, y esto permite que sean aplicaciones portables. Se caracteriza por su enfoque hacia el desarrollo de microservicios lo que permite modularizar aplicaciones y realizar una implementación rápida de nuevos procesos que hagan uso de dichos microservicios.

En este trabajo se ha utilizado Docker como tecnología para encapsular los tres módulos que se han desarrollado. Para realizar el despliegue dentro de la arquitectura se ha llevado a cabo a través de Docker Swarm[vi] y Docker Compose[vii].

Figura 3. 15: Logo Docker

En la Figura 3.16 se puede apreciar el despliegue de cada tecnología.

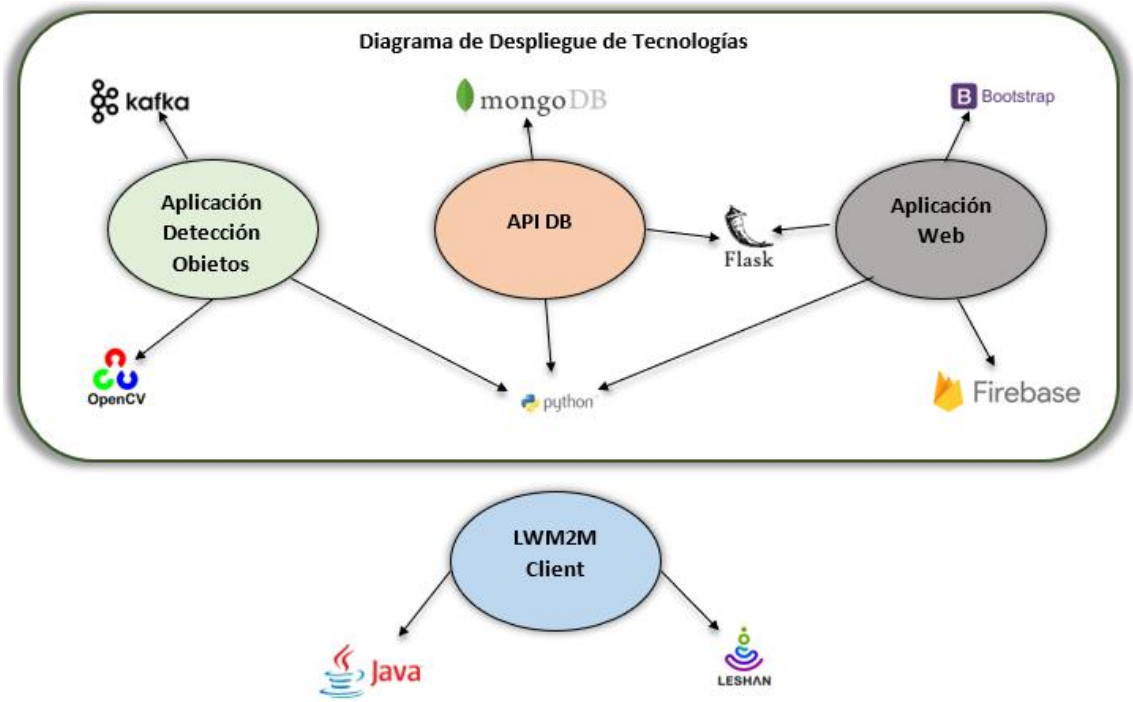

Figura 3. 16: Diagrama de Despliegue de Tecnologías

4

Etapas del desarrollo del trabajo

En este capítulo se procede a explicar las etapas que se han llevado a cabo para la realización del proyecto así como la metodología utilizada para ello. En el apartado de implementación se expondrá de forma extendida la funcionalidad del código de los tres módulos desarrollados.

4.1 Metodologías

Las metodologías software constituyen un proceso disciplinado para poder controlar, estructurar y planificar el desarrollo de software. Existen diversas metodologías que se han ido creando y evolucionando a la par que lo hacía el propio desarrollo software.

Existen dos tipos de metodologías, las metodologías tradicionales y las metodologías ágiles.

Las metodologías tradicionales están más enfocada a la elaboración de mucha documentación, antes de comenzar con la propia implementación, se le asigna una mayor importancia a las tareas de diseño y análisis. Algunas metodologías tradicionales son el *desarrollo en cascada*[viii], *RUP*[ix] (*Rational Unified Process*).

Por su parte, las metodologías ágiles están más enfocada a al resultado del producto. Se caracterizan por una menor cantidad de documentación, ya que estos proyectos sufren constantes cambios en los requisitos que son motivados por la casi

constante interacción entre el cliente y el equipo de desarrollo. Algunas metodologías ágiles existentes son *XP*^{x} (*eXtrem Programming*) y *SCRUM* entre otras.

Para la elaboración de este proyecto se ha decidido usar SCRUM, ejerciendo como cliente tanto el alumno como ambos tutores.

SCRUM es un tipo de metodología ágil que fija un conjunto de roles y prácticas para establecer el desarrollo que se llevará a cabo en el proyecto. Esta metodología se adapta perfectamente a proyectos de pequeña y mediana envergadura.

Entre los roles que establece SCRUM podemos destacar:

- **Scrum Máster** → Es el encargado de llevar a cabo la aplicación de dicha metodología y gestionar los cambios.
- **Product Owner** → Es el representante del cliente, el cual proporciona una visión del proyecto y a su vez la información necesaria para la elaboración de los requisitos.
- **Equipo** → Es el conjunto de personas que se encargan de ejecutar el desarrollo y demás elementos relacionados.

Algunos conceptos importantes de esta metodología son:

- **Sprints** → Periodo de tiempo que tiene una duración entre una y cuatro semanas.
- **Entregables** → Es el resultado final que se obtiene al finalizar el sprint. Suele ser un producto que puede ser ejecutado y mostrado al cliente. El entregable lleva implementadas las funcionalidades de los sprints anteriores y la del sprint actual.

4.2 Análisis

El análisis de un proyecto es una etapa importante en el desarrollo del producto. En esta etapa se sientan las bases del comportamiento y funcionamiento de nuestro sistema, mediante la recopilación de los requisitos, tanto funcionales como no funcionales además de la identificación de los actores principales. Es por ello que una errónea etapa de análisis puede llevar a ambigüedades y contradicciones que dificultarán el desarrollo correcto del proyecto.

4.2.1 Descripción del sistema

Nuestro sistema se compone de tres módulos que se van a incorporar a la arquitectura ya existente (véase en la Figura 4.1, color amarillo) . Con el desarrollo de estos tres módulos se pretende alcanzar el objetivo principal de nuestro proyecto que es poder evaluar el comportamiento de la arquitectura ante una mayor carga de trabajo. Estos módulos, al igual que la arquitectura, se despliegan mediante contenedores Docker en un enjambre (Docker Swarm). Nuestro sistema hace uso de la imagen de MongoDB de Docker y de una API REST que se ha desarrollado para la comunicación entre la aplicación de detección de objetos, la base de datos Mongo y la aplicación web.

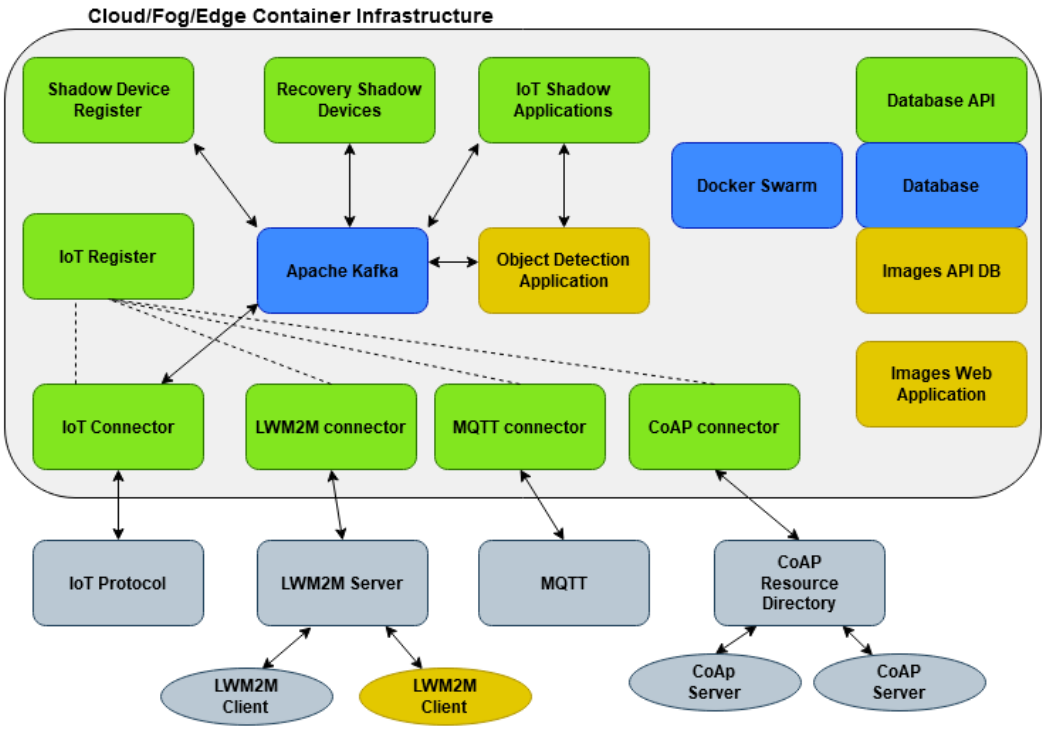

Figura 4. 1: Arquitectura del Sistema con Nuevos Módulos

Cliente Leshan

Este módulo es el encargado de transmitir la imagen y datos sobre ésta al servidor Leshan que se encuentra desplegado en la arquitectura. Leshan como se ha mencionado anteriormente es una biblioteca para poder desarrollar tanto cliente como servidores que se comunicarán mediante el protocolo de comunicación *LWM2M*. Este protocolo, en el lado del cliente, divide a los dispositivos en objetos, y cada objeto tiene

una serie de recursos. Este protocolo permite realizar una conjunto de peticiones sobre un determinado recurso de un objeto. Estas peticiones pueden ser :

- ***READ*** → Esta petición se utiliza para consultar el valor de un determinado recurso.
- ***WRITE*** → Esta petición se utiliza para modificar el valor de un determinado recurso.
- ***EXECUTE*** → Esta petición se utiliza para ejecutar algún tipo de acción sobre el recurso, en nuestro caso la utilizaremos para simular una acción sobre una cámara y tomar una nueva imagen.
- ***OBSERVE*** → Esta petición se utiliza para realizar una observación sobre un recurso cuyo valor va modificándose. Esta acción se usará para transmitir la imagen.
- ***DELETE OBSERVATION*** → Esta petición se utiliza para cancelar una observación.

Por lo tanto, para lograr nuestro objetivo, es necesario desarrollar un objeto que tenga diversos recursos a los que se puedan realizar estas peticiones para poder obtener la información de una imagen y enviarla a la arquitectura.

Aplicación de Detección de Objetos

Este módulo se encargará de procesar los datos que le suministra la arquitectura a través de un consumidor de Apache Kafka. Para poder recibir los datos, primero debe, obligatoriamente, realizar dos tipos de peticiones. Estas dos peticiones son obligatorias para poder permitir a la arquitectura de detección de fallos comprender el funcionamiento de nuestra aplicación y poder replicarlo en caso de fallo.

La primera petición consiste en mostrar interés al módulo *iotshadowapplications* sobre un determinado recurso del cliente leshan.

La segunda petición consistirá en conocer el valor o en realizar una acción sobre el recurso que anteriormente hemos mostrado interés. Esta petición también debe ser solicitada al módulo *iotshadowapplications*. Esta interacción se puede observar más adelante en la Figura 4.12.

Una vez se han recibido los datos correspondiente a una imagen, esta aplicación a través del algoritmo de detección de objetos *YOLO*, procesará la imagen y haciendo uso de una API REST, la almacenará en una base de datos para su posterior visualización.

Aplicación Web

Este módulo es una pequeña aplicación web que nos permitirá ver el resultado del procesamiento de la imagen. A través de una petición a una API REST que se ha desarrollado para establecer comunicación con nuestra base de datos, se solicita la imagen procesada. El acceso a nuestra página web se hará a través de la tecnología oAuth 2.0 a través de una cuenta de Google. Esta implementación de acceso se ayuda de la plataforma Firebase.

4.2.2 Requisitos

Requisitos Funcionales

Los requisitos funcionales definen una funcionalidad del sistema. A continuación vamos a citar los requisitos funcionales de cada módulo desarrollado.

Cliente Leshan

RF-01. Reiniciar dispositivo.

RF-02. Consultar Manufacturación.

RF-03. Consultar Número de modelo.

RF-04. Consultar número serie.

RF-05. Consultar versión firmware.

RF-06. Consultar tiempo del dispositivo.

RF-07. Consultar UTC.

RF-08. Modificar UTC.

RF-09. Consultar zona horaria.

RF-10. Modificar zona horaria.

RF-11. Consultar Modos bindings soportados.

RF-12. Consultar tipo de dispositivo.

RF-13. Consultar versión software.

RF-14. Consultar código de error.

RF-15. Consultar estatus de batería.

RF-16. Tomar imagen.

RF-17. Consultar ID imagen.

RF-18. Consultar nombre imagen.

RF-19. Consultar fecha imagen.

RF-20. Consultar número de peticiones totales.

RF-21. Consultar peticiones realizadas.

RF-22. Observar Fragmento imagen.

Aplicación de Detección de Objetos

RF-23. Mostrar Interés sobre un recurso.

RF-24. Realizar acción sobre un recurso.

RF-25. Detectar objetos.

RF-26. Almacenar imagen.

Aplicación Web

RF-27. Inicio de Sesión.

RF-28. Cierre de Sesión.

RF-29. Registro de usuario.

RF-30. Eliminar cuenta.

RF-31. Consultar imágenes.

RF-32. Seleccionar una imagen.

RF-33. Mostrar Imagen.

Requisitos No Funcionales

Los requisitos no funcionales se refieren a las propiedades generales o restricciones sobre el sistema que se esté desarrollando. Ejemplos de requisitos no funcionales hacen referencia a atributos como eficiencia, usabilidad, seguridad…

Los requisitos no funcionales de nuestro sistema son:

RNF-01. Tamaño máximo de bytes enviados en cada lectura es de 8154.

RNF-02. Tiempo de reinicio del dispositivo < 500 milisegundos.

RNF-03. Sólo existirá una instancia del objeto *imageObject*.

RNF-04. El tiempo de actualización de datos en la petición observe será de 100 milisegundos.

RNF-05. La aplicación de procesado de imagen se comunicará con la base de datos a través de una API REST.

RNF-06. La identificación del usuario en la aplicación web será a través de la tecnología oAuth 2.0 .

RNF-07. El registro de usuario en la aplicación web será a través de la tecnología oAuth 2.0 .

RNF-08. La aplicación web se comunicará con la base de datos a través de una API REST.

RNF-09. Los módulos serán desplegados en Docker y Docker Swarm.

RNF-10. Para la comunicación interna se usará Apache Kafka.

RNF-11. El formato de los mensajes en las comunicaciones será JSON.

4.3 Especificación

La especificación de un proyecto o sistema software, consiste en la definición exhaustiva del comportamiento del sistema que se procede a desarrollar.

En este apartado se realizará la especificación del sistema a través de la definición y descripción de los casos de uso de los requisitos definidos en el apartado de *Análisis* y el modelo de dominio.

4.3.1 Modelo del dominio

El modelo de dominio es una representación o especificación de un sistema, desde un determinado punto de vista y con un objetivo concreto. Los modelos han de ser abstractos, comprensibles, precisos, predictivos y baratos y se utilizan para especificar o comprender el sistema, razonar sobre el sistema y poder validarlo y como guía para la implementación.

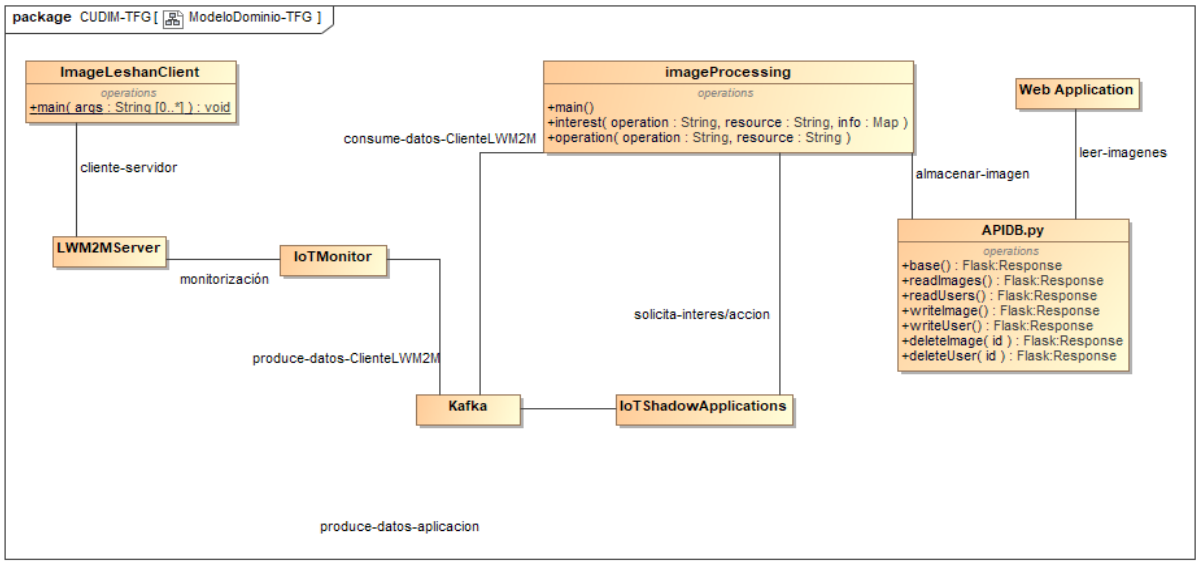

Figura 4. 2 Modelo del Dominio

4.3.2 Casos de Uso

Los casos de uso corresponden con la descripción de una acción o actividad llevada a cabo entre un actor (humano o máquina) y un sistema.

Diagramas de casos de uso

- *Casos de Uso Cliente Leshan*

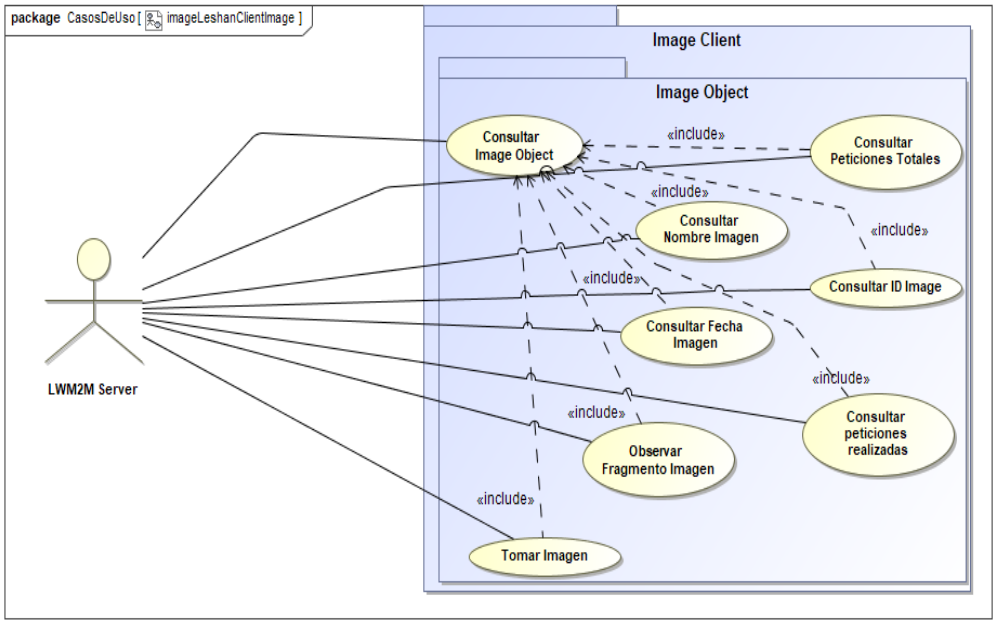

Figura 4. 3: Casos de Uso Cliente Leshan (Image Object)

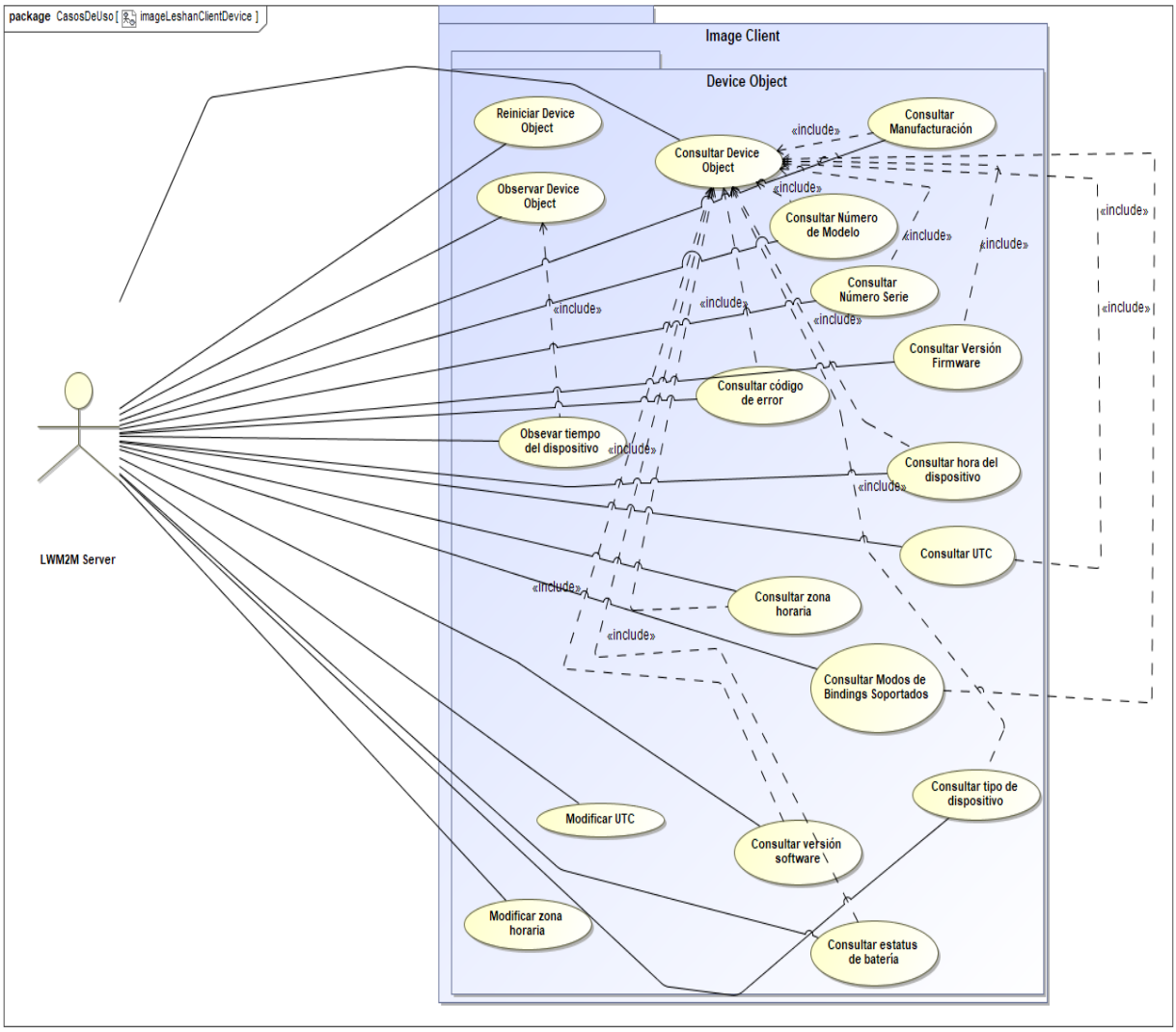

Figura 4. 4: Casos de Uso Cliente Leshan (Device Object)

- *Casos de Uso Aplicación de Detección de Objetos*

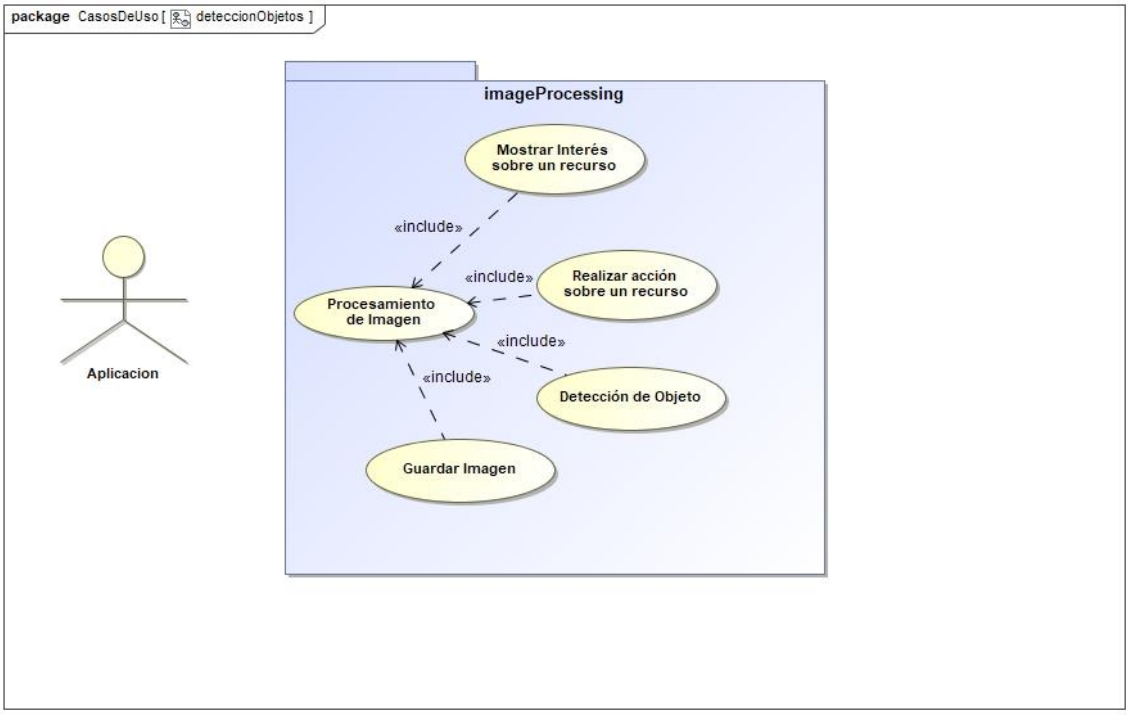

Figura 4.5 : Casos de Uso Aplicación de Detección de Objetos

- *Casos de Uso Aplicación Web*

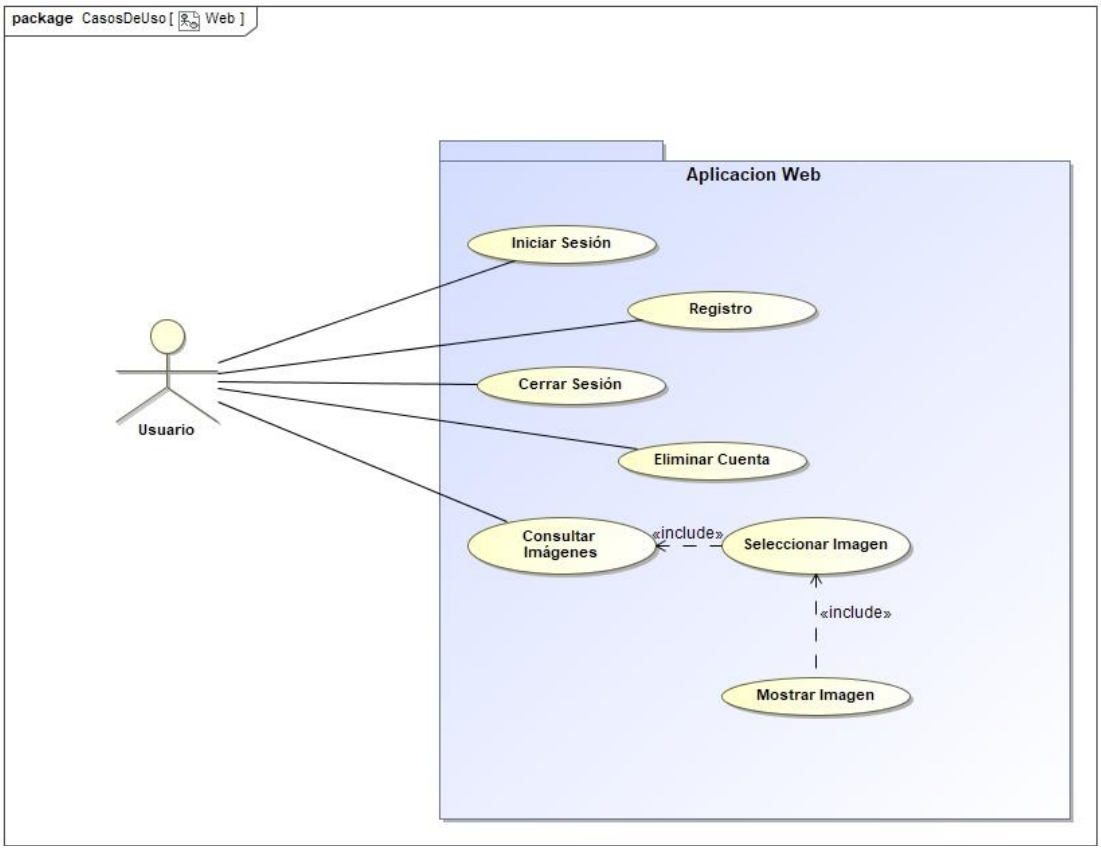

Figura 4.6 : Casos de Uso Aplicación Web

Descripción de casos de uso

En este apartado se procede a la descripción textual de los casos de uso más representativos.

- *Casos de uso de Cliente Leshan*

ID	RF-16
Título	Tomar imagen.
Descripción	El cliente actualizará el valor de la imagen actual.
Pre-condición	El cliente está registrado en el servido LWM2M. El cliente está encendido. El cliente está operativo.
Post-condición (Éxito)	El cliente está operativo.
Post-condición (Error)	El cliente notifica un error.
Escenario principal	
1. El cliente recibe una petición EXECUTE con el id del recurso tomar imagen. 2. El cliente carga una nueva imagen. 3. El cliente notifica éxito en la operación EXECUTE. 4. El cliente está encendido.	
Escenario alternativo	
2.b Se produce un error al cargar la imagen. 2.b.1 Se produce una excepción en el cliente. 2.b.2 El cliente notifica un error.	

ID	RF-17
Título	Consultar ID imagen.
Descripción	El cliente devolverá el id de la actual imagen.
Pre-condición	El cliente está registrado en el servido LWM2M. El cliente está encendido. El cliente está operativo.
Post-condición (Éxito)	El cliente está operativo.
Post-condición (Error)	
Escenario principal	
1. El cliente recibe la petición READ con el id del recurso id imagen. 2. El cliente devuelve el valor del id de la actual imagen. 3. El cliente notifica éxito en la operación READ. 4. El cliente se mantiene a la espera de nuevas peticiones.	
Escenario alternativo	

ID	RF-18
Título	Consultar nombre imagen.
Descripción	El cliente devolverá el nombre de la actual imagen.
Pre-condición	El cliente está registrado en el servido LWM2M. El cliente está encendido. El cliente está operativo.
Post-condición (Éxito)	El cliente está operativo.
Post-condición (Error)	
Escenario principal	
1. El cliente recibe la petición READ con el id del recurso nombre imagen. 2. El cliente devuelve el valor del nombre de la imagen. 3. El cliente notifica éxito en la operación READ. 4. El cliente se mantiene a la espera de nuevas peticiones.	
Escenario alternativo	

ID	RF-19
Título	Consultar fecha imagen.
Descripción	El cliente retornará la fecha de la actual imagen.
Pre-condición	El cliente está registrado en el servido LWM2M. El cliente está encendido. El cliente está operativo.
Post-condición (Éxito)	El cliente está operativo.
Post-condición (Error)	
Escenario principal	
1. El cliente recibe la petición READ con el id del recurso fecha imagen. 2. El cliente devuelve el valor de la fecha de la imagen. 3. El cliente notifica éxito en la operación READ. 4. El cliente se mantiene a la espera de nuevas peticiones.	
Escenario alternativo	

ID	RF-20
Título	Consultar número de peticiones totales.
Descripción	El cliente retornará el número total de peticiones que deben realizarse para obtener la imagen completa.
Pre-condición	El cliente está registrado en el servido LWM2M. El cliente está encendido. El cliente está operativo.
Post-condición (Éxito)	El cliente está operativo.
Post-condición (Error)	
Escenario principal	
1. El cliente recibe la petición READ con el id del recurso estatus de la batería. 2. El cliente devuelve el valor del estatus de la batería. 3. El cliente notifica éxito en la operación READ. 4. El cliente se mantiene a la espera de nuevas peticiones.	
Escenario alternativo	

ID	RF-21
Título	Consultar peticiones realizadas.
Descripción	El cliente retornará el número total de peticiones que se han realizado hasta el momento.
Pre-condición	El cliente está registrado en el servido LWM2M. El cliente está encendido. El cliente está operativo.
Post-condición (Éxito)	El cliente está operativo.
Post-condición (Error)	
Escenario principal	
1. El cliente recibe la petición READ con el id del recurso peticiones actuales. 2. El cliente devuelve el valor de las peticiones actuales. 3. El cliente notifica éxito en la operación READ. 4. El cliente se mantiene a la espera de nuevas peticiones.	
Escenario alternativo	

ID	RF-22
Título	Observar Fragmento imagen.
Descripción	El cliente devolverá cada cierto tiempo un nuevo fragmento de la imagen.
Pre-condición	El cliente está registrado en el servido LWM2M. El cliente está encendido. El cliente está operativo.
Post-condición (Éxito)	El cliente está operativo.
Post-condición (Error)	Parar observación.
Escenario principal	
1. El cliente recibe la petición OBSERVE con el id del recurso bytes imagen. 2. El cliente devuelve el valor de bytes imagen correspondiente a la petición actual. 3. El cliente se mantiene a la espera de nuevas peticiones. 4. Pasados 2000 ms, vuelve a paso 2.	
Escenario alternativo	
4.b El cliente recibe la petición Cancel_Observation. 4.b.1 El cliente no devuelve el valor de bytes imagen correspondiente a la petición actual. 4.b.2 El cliente se mantiene a la espera de nuevas peticiones.	

Casos de uso de Aplicación Detección de Objetos:

ID	RF-23
Título	Mostrar Interés sobre un recurso.
Descripción	La aplicación mediante una petición al módulo *iotshadowapplications*, mostrará interés en realizar una determinada acción sobre un recurso de un objeto del cliente LWM2M.
Pre-condición	El cliente está registrado en la arquitectura. El cliente está encendido. El cliente está operativo.
Post-condición (Éxito)	Se ha mostrado interés correctamente.
Escenario principal	
1. La aplicación realiza una petición GET de interés sobre un recurso al módulo *iotshadowapplications*. 2. La aplicación recibe como respuesta un código de estado 200.	

ID	RF-24
Título	Realizar acción sobre un recurso.
Descripción	La aplicación mediante una petición al módulo *iotshadowapplications*, solicitará realizar una determinada acción sobre un recurso de un objeto del cliente LWM2M.
Pre-condición	El cliente está registrado en la arquitectura. El cliente está encendido. El cliente está operativo.
Post-condición (Éxito)	Se ha mostrado interés correctamente.
Escenario principal	
1. La aplicación realiza una petición GET de realización de acción *OBSERVE* sobre un recurso al módulo *iotshadowapplications*. 2. La aplicación recibe como respuesta un código de estado 200. 3. La aplicación se suscribe al topic de Kafka recibido. 4. La aplicación lee los datos desde Kafka. 5. La aplicación termina de leer los datos.	

ID	RF-25
Título	Detectar objetos
Descripción	La aplicación realizará un análisis para la detección de objetos con el algoritmo Yolo.
Pre-condición	La aplicación ha recibido la imagen.
Post-condición (Éxito)	La imagen ha sido analizada.
Escenario principal	
1. La aplicación realiza la detección de objetos de la imagen recibida.	
Escenario alternativo	

ID	RF-26
Título	Almacenar imagen.
Descripción	La aplicación guardará en la base de datos información relevante sobre la imagen procesada.
Pre-condición	La imagen ha sido procesada.
Post-condición (Éxito)	La información sobre la imagen está almacenada en la base de datos.
Post-condición (Error)	
Escenario principal	
1. La aplicación envía una petición a la API REST de la base de datos para almacenar: • Nombre de imagen • Fecha de la imagen • Bytes de la imagen 2. La aplicación recibe notificación de éxito en la operación.	
Escenario alternativo	

Casos de uso de Aplicación Web:

ID	RF-27
Título	Inicio de Sesión.
Descripción	El usuario podrá ingresar en la aplicación web usando su cuenta de Google en la página de inicio.
Pre-condición	El usuario está registrado en el sistema. El usuario está en la página principal.
Post-condición (Éxito)	El usuario identificado se encuentra en la página principal.
Post-condición (Error)	El usuario no se identifica.
Escenario principal	
1. El usuario pulsa el botón de iniciar sesión. 2. El sistema deriva a la página de sesión de Google. 3. El usuario introduce sus credenciales. 4. El usuario pulsa el botón de iniciar sesión. 5. El sistema muestra la página de inicio.	
Escenario alternativo	
4.b El usuario no está registrado previamente. 4.b.1 El sistema lo deriva a la página principal. 4.c El usuario cancela la acción. 4.c.1 El sistema deriva al usuario a la página principal.	

ID	RF-28
Título	Cierre de Sesión.
Descripción	El usuario podrá desconectarse de la aplicación web.
Pre-condición	El usuario está conectado.
Post-condición (Éxito)	El usuario deja de estar conectado. El usuario se encuentra en la página de inicio.
Post-condición (Error)	
Escenario principal	
1. El usuario pulsa el botón cerrar sesión. 2. El sistema desconecta al usuario. 3. El sistema redirige al usuario a la página de inicio.	
Escenario alternativo	

ID	RF-29
Título	Registro de usuario.
Descripción	El usuario podrá darse de alta en la aplicación web utilizando su cuenta de Google.
Pre-condición	El usuario no está registrado. El usuario se encuentra en la página principal
Post-condición (Éxito)	El usuario se encuentra registrado.
Post-condición (Error)	El usuario no se ha registrado.
Escenario principal	
1. El usuario pulsa el botón iniciar sesión. 2. El sistema redirige al usuario a la pantalla de inicio de sesión de Google. 3. El usuario introduce sus credenciales. 4. El usuario inicia sesión en Google. 5. El sistema comprueba si no está registrado anteriormente. 6. El sistema almacena el correo electrónico, nombre y apellidos de su cuenta de Google. 7. El sistema redirige al usuario a la página de inicio.	
Escenario alternativo	
4.b. El usuario cancela la operación. 4.b.1 El sistema redirige al usuario a la página principal. 5.b. El usuario ya está registrado. 5.b.1 El sistema redirige al usuario a la página de inicio.	

ID	RF-30
Título	Eliminar cuenta.
Descripción	El usuario podrá suprimir su perfil de la aplicación web.
Pre-condición	El usuario ha iniciado sesión. El usuario está en su perfil.
Post-condición (Éxito)	El sistema elimina el perfil del usuario.
Post-condición (Error)	El sistema no elimina el perfil del usuario.
Escenario principal	
1. El usuario pulsa el botón de eliminar perfil. 2. El sistema elimina el perfil del usuario. 3. El sistema redirige al usuario a la página de inicio.	
Escenario alternativo	
3.b Se produce un error durante el borrado. 3.b.1 El sistema redirige al usuario a la página perfil.	

ID	RF-31
Título	Consultar Imágenes.
Descripción	El usuario podrá visualizar una lista de las imágenes que se encuentran en el sistema.
Pre-condición	El usuario ha iniciado sesión. El usuario se encuentra en la página principal.
Post-condición (Éxito)	El sistema muestra un listado de las imágenes del sistema.
Post-condición (Error)	
Escenario principal	
1. El sistema muestra las imágenes almacenadas en el sistema.	
Escenario alternativo	

ID	RF-32
Título	Seleccionar imagen
Descripción	El usuario podrá seleccionar una imagen para su posterior visualización.
Pre-condición	El usuario ha iniciado sesión. El usuario se encuentra en la página principal.
Post-condición (Éxito)	El sistema deriva al usuario a la página de visualización.
Post-condición (Error)	
Escenario principal	
1. El usuario selecciona una imagen a su preferencia. 2. El sistema deriva al usuario a una nueva página.	
Escenario alternativo	

ID	RF-33
Título	Mostrar imagen.
Descripción	El sistema visualizará la imagen seleccionada.
Pre-condición	El usuario ha seleccionado una imagen.
Post-condición (Éxito)	El sistema visualiza la imagen seleccionada.
Post-condición (Error)	
Escenario principal	
1. El sistema visualiza la imagen. 2. El usuario pulsa el botón volver. 3. El sistema redirige al usuario a la página principal.	
Escenario alternativo	
3.b El usuario no pulsa el botón volver. 3.b.1 El sistema se mantiene a la espera hasta que pulsa el botón volver.	

4.3.3 Casos de prueba

Los casos de prueba corresponden con la descripción de un conjunto de condiciones o variables bajo las cuales se determina si un sistema software realiza su funcionalidad de manera satisfactoria.

Descripción de casos de prueba

En este apartado se procede a la descripción textual de los casos de prueba más representativos.

- *Casos de prueba de Cliente Leshan*

  Los casos de pruebas del sistema embebido han seguido la estructura del documento *Enabler Test Specification for LightweightM2M.*[xi]

ID Requisito	RF-16
Descripción	Comprobar si es posible tomar una imagen de forma remota.
Pre-condición	El dispositivo está registrado en el servidor LWM2M. El dispositivo está encendido.
Proceso de Prueba	
1. Una operación EXECUTE es enviada por el servidor al recurso TAKE_PHOTO del objeto Image Data. *Flujo Normal:* a. El servidor realiza la petición EXECUTE (COAP Post) en el recurso TAKE_PHOTO del objeto Image Data. b. El servidor recibe mensaje de éxito por parte del cliente	
Criterio de éxito	
1. El dispositivo toma la imagen de forma satisfactoria.	

ID-Requisito	RF-17
Descripción	Consulta en el cliente de la siguiente información: • ID de la imagen.
Pre-condición	El dispositivo está registrado en el servidor LWM2M.
Proceso de Prueba	
1. Una petición READ desde el servidor sobre el recurso IMAGE_ID ha sido recibida por el cliente. *Flujo Normal:* a. READ (COAP GET) sobre el recurso IMAGE_ID del objeto Image Data. b. El servidor recibe mensaje de éxito y la información requerida.	
Criterio de éxito	
1. El servidor ha recibido la información solicitada y muestra los siguientes datos al usuario si es posible: • ID de la imagen.	

ID-Requisito	RF-18
Descripción	Consulta en el cliente de la siguiente información: •Nombre de la imagen.
Pre-condición	El dispositivo está registrado en el servidor LWM2M.
Proceso de Prueba	
1. Una petición READ desde el servidor sobre el recurso IMAGE_NAME ha sido recibida por el cliente. *Flujo Normal:* a. READ (COAP GET) sobre el recurso IMAGE_NAME de Image Data. b. El servidor recibe mensaje de éxito y la información requerida.	
Criterio de éxito	
1.El servidor ha recibido la información solicitada y muestra los siguientes datos al usuario si es posible: • Nombre de la imagen.	

ID-Requisito	RF-19
Descripción	Consulta en el cliente de la siguiente información: • Fecha de la imagen.
Pre-condición	El dispositivo está registrado en el servidor LWM2M.
Proceso de Prueba	
1. Una petición READ desde el servidor sobre el recurso IMAGE_TIME ha sido recibida por el cliente. *Flujo Normal:* a. READ (COAP GET) sobre el recurso IMAGE_TIME de Image Data. b. El servidor recibe mensaje de éxito y la información requerida.	
Criterio de éxito	
1. El servidor ha recibido la información solicitada y muestra los siguientes datos al usuario si es posible: • Fecha de la imagen.	

ID-Requisito	RF-20
Descripción	Consulta en el cliente de la siguiente información: • Peticiones totales.
Pre-condición	El dispositivo está registrado en el servidor LWM2M.
Proceso de Prueba	
1. Una petición READ desde el servidor sobre el recurso IMAGE_NUMBER_OF_REQUEST ha sido recibida por el cliente. *Flujo Normal*: a. READ (COAP GET) sobre el recurso IMAGE_NUMBER_OF_REQUEST de Image Data. b. El servidor recibe mensaje de éxito y la información requerida.	
Criterio de éxito	
1. El servidor ha recibido la información solicitada y muestra los siguientes datos al usuario si es posible: • Peticiones totales.	

ID-Requisito	RF-21
Descripción	Consulta en el cliente de la siguiente información: • Peticiones realizadas.
Pre-condición	El dispositivo está registrado en el servidor LWM2M.
Proceso de Prueba	
1. Una petición READ desde el servidor sobre el recurso IMAGE_ACTUAL_REQUEST ha sido recibida por el cliente. *Flujo Normal:* a. READ (COAP GET) sobre el recurso IMAGE_ACTUAL_REQUEST de Image Data. b. El servidor recibe mensaje de éxito y la información requerida.	
Criterio de éxito	
1. El servidor ha recibido la información solicitada y muestra los siguientes datos al usuario si es posible: • Peticiones realizadas.	

ID-Requisito	RF-22
Descripción	Envío de la política de observación al recurso BYTES del objeto Image Data.
Pre-condición	El dispositivo está registrado en el servidor LWM2M. El dispositivo está encendido. El dispositivo está operativo.
Proceso de Prueba	
1. El servidor establece una relación de Observación con el cliente para adquirir notificaciones sobre: • Bytes. *Flujo Normal:* 1. El servidor se comunica con el dispositivo en un periodo de tiempo. 2. El servidor envía un mensaje OBSERVE (COAP Observe Option) para activar el flujo de comunicación. 3. El cliente devuelve la información solicitada con un mensaje NOTIFY (COAP responses).	
Criterio de éxito	
1. El servidor ha recibido la información solicitada y muestra los siguientes datos al usuario si es posible: • IMAGE_BYTES.	

- *Casos de prueba de Aplicación de Detección de Objetos*

ID-Requisito	RF-23
Descripción	Envío de una petición get de interés sobre un recurso.
Pre-condición	El dispositivo está registrado en la arquitectura. El dispositivo está encendido. El dispositivo está operativo.
Proceso de Prueba	
1. La aplicación recibe un código de estado 200. *Flujo Normal:* 1. La aplicación genera una petición http get solicitando interés sobre un determinado recurso. 2. El módulo *iotshadowapplications* devuelve una respuesta a la petición.	
Criterio de éxito	
1. La respuesta contiene un código de estado 200.	

ID-Requisito	RF-24
Descripción	Envío de una petición get de realizar acción sobre un recurso.
Pre-condición	El dispositivo está registrado en la arquitectura. El dispositivo está encendido. El dispositivo está operativo.
Proceso de Prueba	
1. La aplicación recibe los datos correspondientes al recurso solicitado. *Flujo Normal:* 1. La aplicación genera una petición http get solicitando realizar una acción sobre un determinado recurso. 2. El módulo *iotshadowapplications* devuelve una respuesta a la petición. 3. La aplicación se suscribe al topic de Kafka proporcionado en la respuesta.	
Criterio de éxito	
1. La aplicación recibe los datos solicitados.	

ID-Requisito	RF-26
Descripción	Comprobar si se ha almacenado los datos en la base de datos.
Pre-condición	La aplicación ha analizado la imagen.
Proceso de Prueba	
1. La aplicación envía una petición a la API de la base de datos. *Flujo Normal:* a. La aplicación envía los datos en formato JSON. b. La API realiza la consulta de inserción en la base de datos.	
Criterio de éxito	
1. Se han insertado correctamente los datos.	

- *Casos de prueba de Aplicación Web*

Para la descripción de los casos de prueba se ha usado la sintaxis de Gherkin[xii]

ID-Requisito	RF-27

Feature: Iniciar sesión.

Los usuarios podrán conectarse a la aplicación a través de su cuenta de Google.

Background:

Given un usuario llamado “Juan”.

And Juan está registrado.

And un usuario llamado “Pepe”.

And Pepe no está registrado.

Scenario: Juan inicia sesión.

When Juan intente iniciar sesión.

Then Debe ingresar en la aplicación.

Scenario: Pepe no inicia sesión.

When Pepe intente iniciar sesión.

Then No debe ingresar en la aplicación.

ID-Requisito	RF-28

Feature: Cerrar Sesión.

Los usuarios podrán desconectarse de la aplicación.

Background:

Given un usuario llamado “Juan”.

And Juan está registrado.

And Juan ha iniciado sesión.

Scenario: Juan cierra sesión.

When Juan intente cerrar sesión.

Then Juan debe abandonar la aplicación.

ID-Requisito	RF-29

Feature: Registro de usuario.

Los usuarios darse de alta en la aplicación usando su cuenta de Google.

Background:

Given un usuario llamado “Juan”.
And Juan está registrado.
And un usuario llamado “Pepe”.
And Pepe no está registrado.

Scenario: Pepe se registra.

When Pepe intente registrarse.
Then Debe ingresar en la aplicación.

Scenario: Pepe no se registra.

When Pepe intente registrarse.
And Pepe cancele la operación.
Then Pepe no debe ingresar en la aplicación.

Scenario: Juan no se registra.

When Juan intente registrarse
Then Debe ingresar en la aplicación

ID-Requisito	RF-30

Feature: Eliminar Cuenta.

Los usuarios podrán eliminar su perfil de la aplicación.

Background:

Given un usuario llamado “Juan”.

And Juan está registrado.

And Juan ha iniciado sesión.

And Juan está en la página “Mi Perfil”.

And Juan pulsa el botón Eliminar.

Scenario: Juan elimina su cuenta.

When Juan pulsa el botón “Borrar Perfil”.

Then Juan debe no estar registrado.

Scenario: Juan no elimina su cuenta.

When Juan pulsa el botón “Borrar Perfil”.

Then Juan debe siguir registrado .

ID-Requisito	RF-31

Feature: Consultar imágenes.

Los usuarios deben poder visualizar las imágenes del Sistema.

Background:

Given un usuario llamado “Juan”.

And Juan está registrado.

And Juan ha iniciado sesión.

Scenario: Juan visualiza las imágenes.

When Juan se encuentre en la página principal.

Then Juan debe ver las imágenes del sistema.

ID-Requisito	RF-32

Feature: Seleccionar Imagen.

Los usuarios deben poder seleccionar las imágenes del Sistema.

Background:

Given un usuario llamado “Juan”.

And Juan está registrado.

And Juan ha iniciado sesión.

Scenario: Juan selecciona una imagen.

When Juan se encuentre en la página principal.

And Juan seleccione una imagen.

Then Juan debe ser redirigido a una nueva página.

ID-Requisito	RF-33

Feature: Mostar imagen.

Los usuarios deben poder visualizar la imagen seleccionada.

Background:

Given un usuario llamado “Juan”.

And Juan está registrado.

And Juan ha iniciado sesión.

And Juan ha seleccionado una imagen.

Scenario: Juan visualiza la imágenes.

When Juan se encuentre en la página de visualización.

Then Juan debe ver la imagen.

Scenario: Juan vuelve a la página principal.

When Juan se encuentre en la página de visualización.

And Juan pulsa el botón “Volver”.

Then Juan debe retornar a la página principal.

4.3.4 Diagramas de Secuencia

El diagrama de secuencia es un diagrama que sirve para modelar las interacciones entre los objetos de una aplicación a lo largo del tiempo. En nuestro caso modelaremos la interacción entre los componentes de nuestro sistema.

A continuación se muestran aquellos diagramas que se han considerado más relevantes.

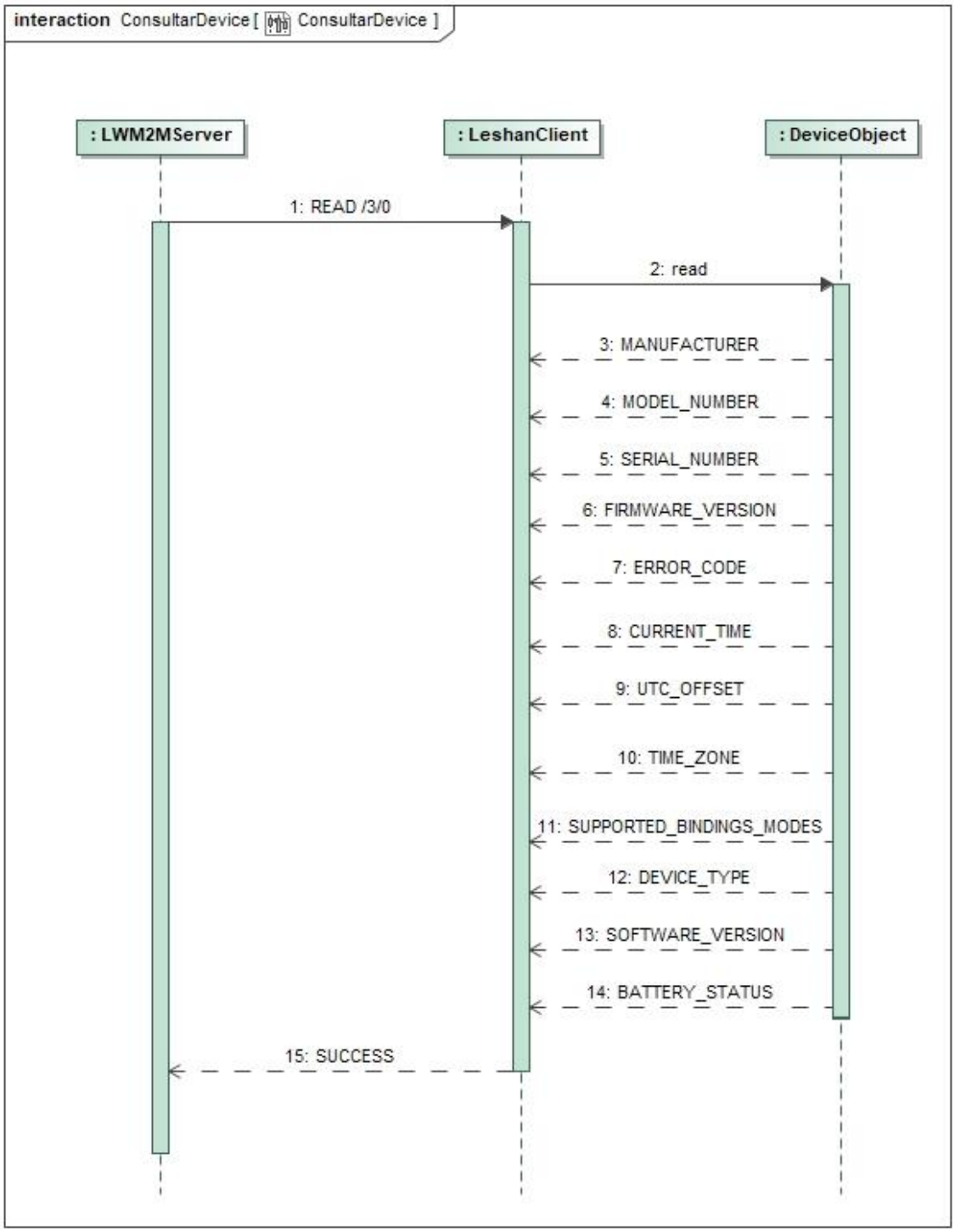

Figura 4.7 : Operación READ sobre el objeto Device

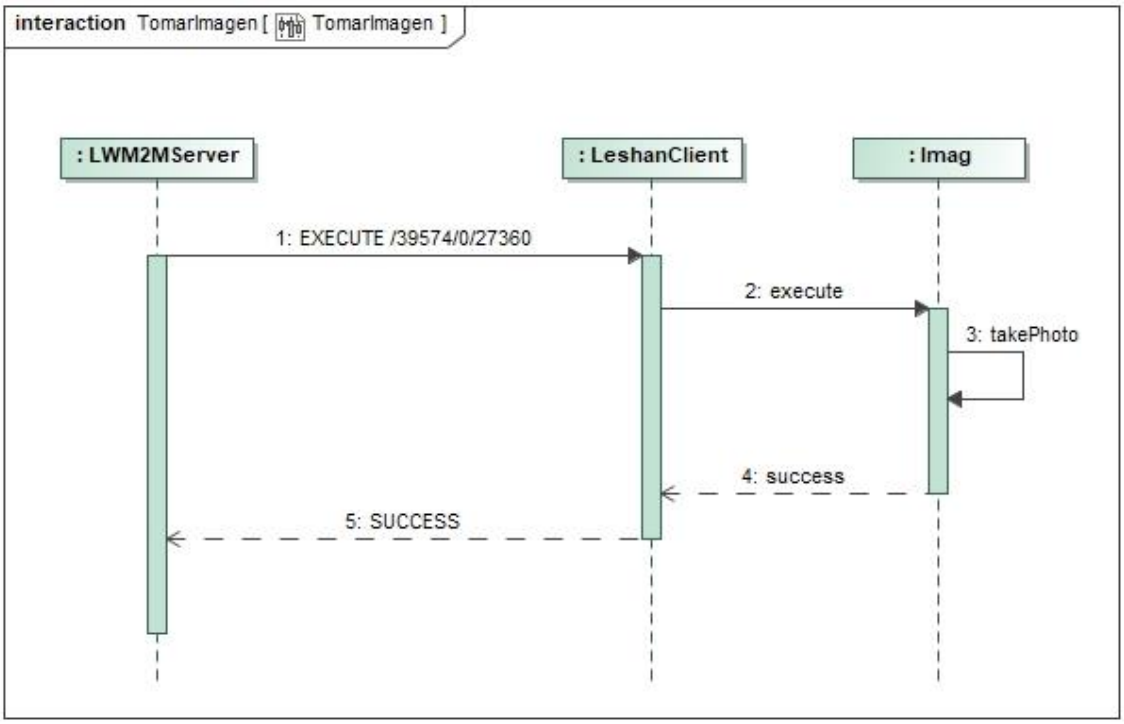

Figura 4.8 : Operación EXECUTE sobre el recurso Take Photo del objeto Image

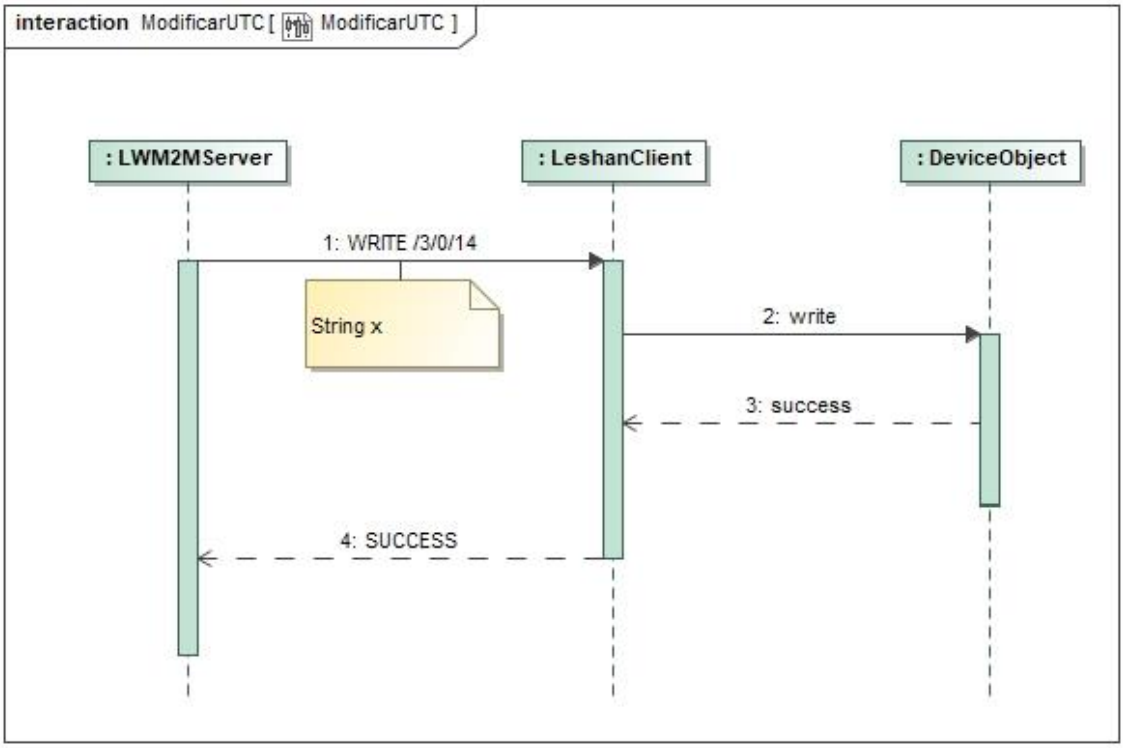

Figura 4.9 : Operación WRITE sobre el recurso UTC del objeto Device

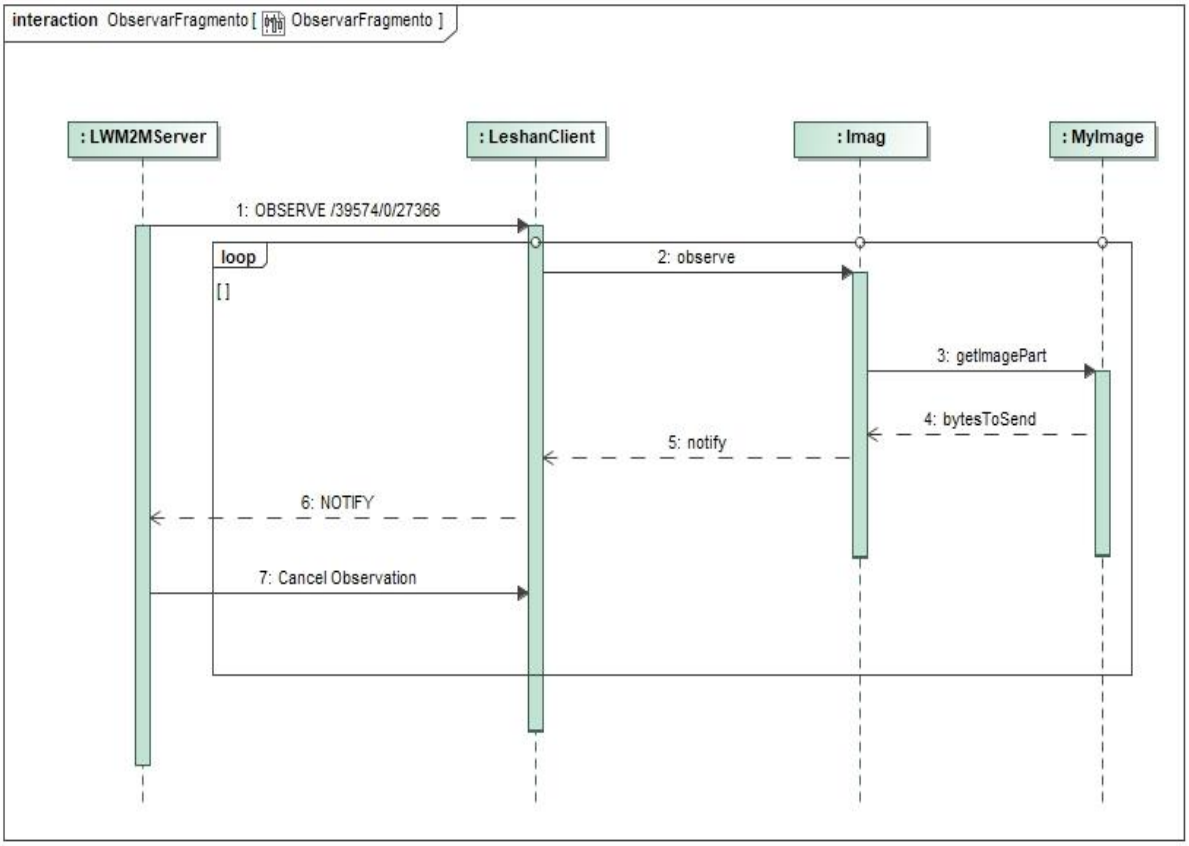

Figura 4.10 : Operación OBSERVE sobre el recurso Bytes del objeto Image

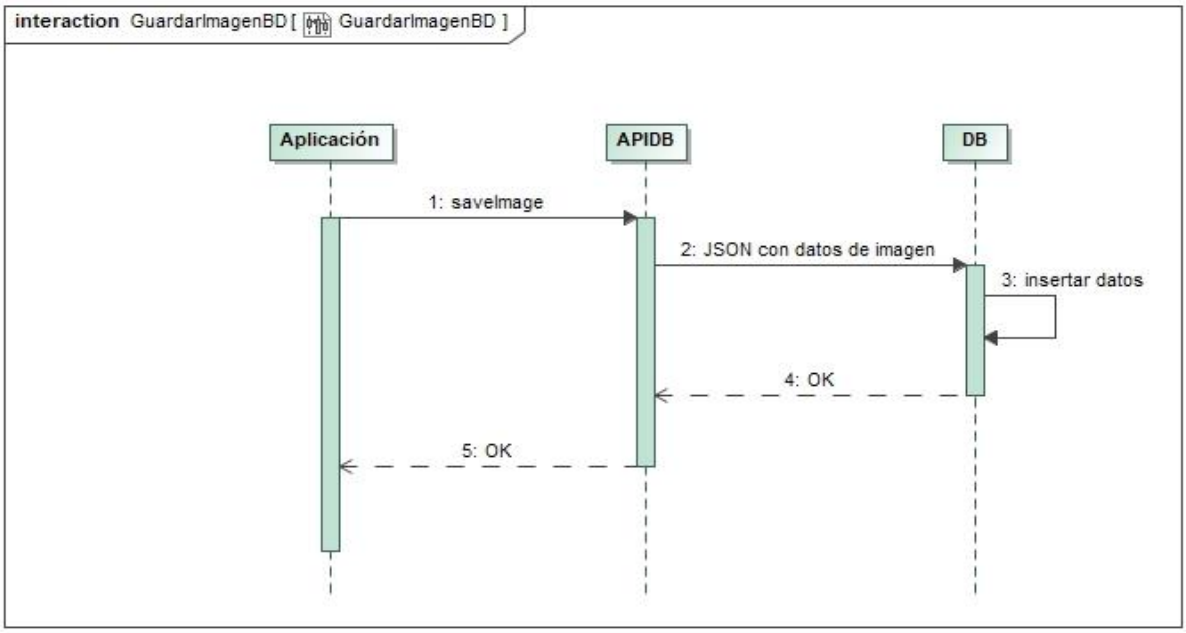

Figura 4.11 : Guardar Imagen en la base de datos

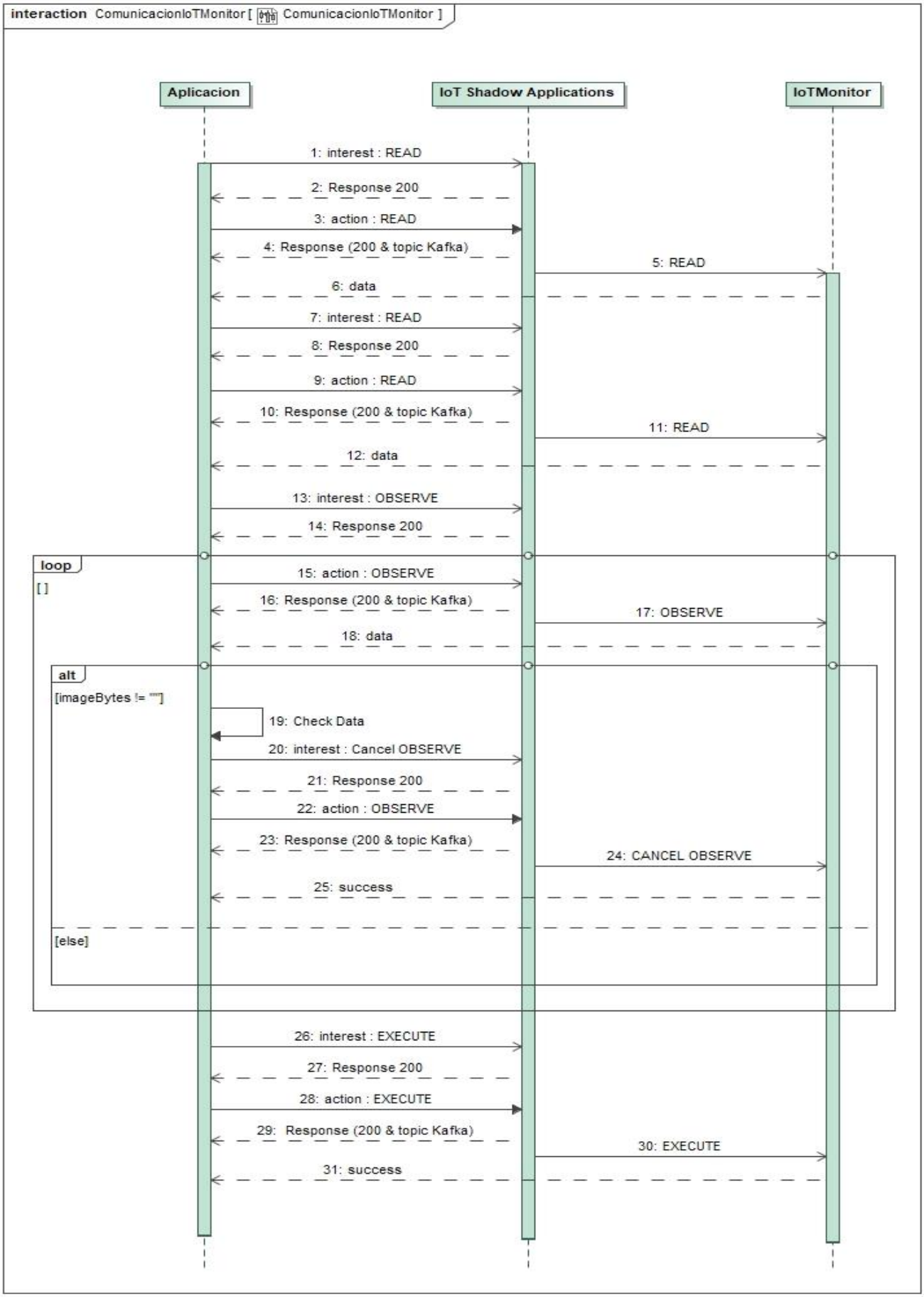

Figura 4.12 : Comunicación con IoTMonitor para obtener datos de imagen

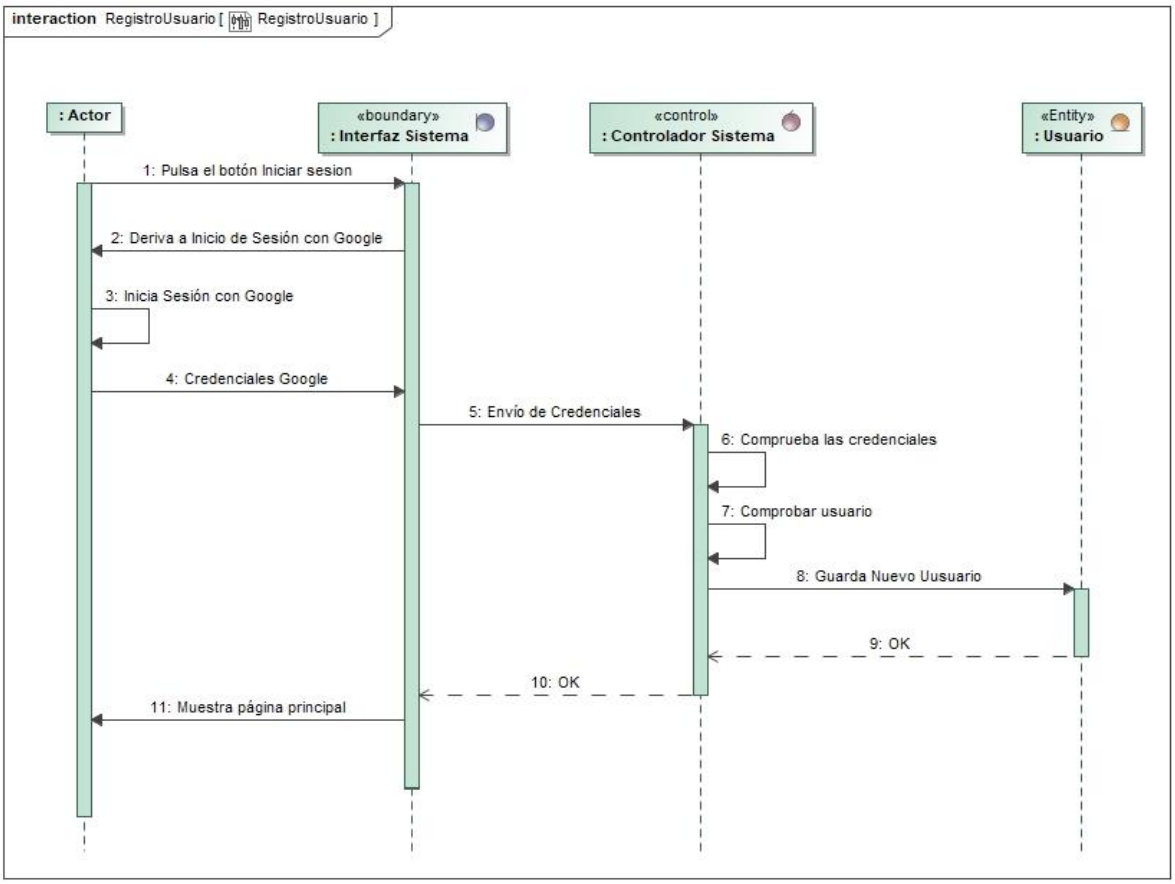

Figura 4.13 : Registro de Usuario

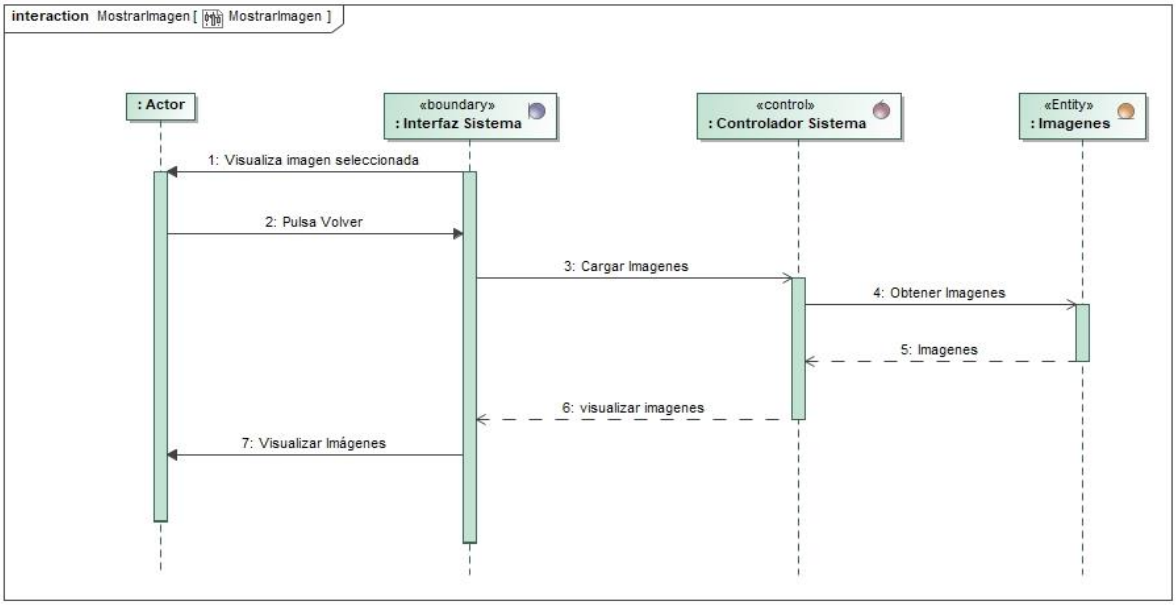

Figura 4.14 : Mostrar Imagen

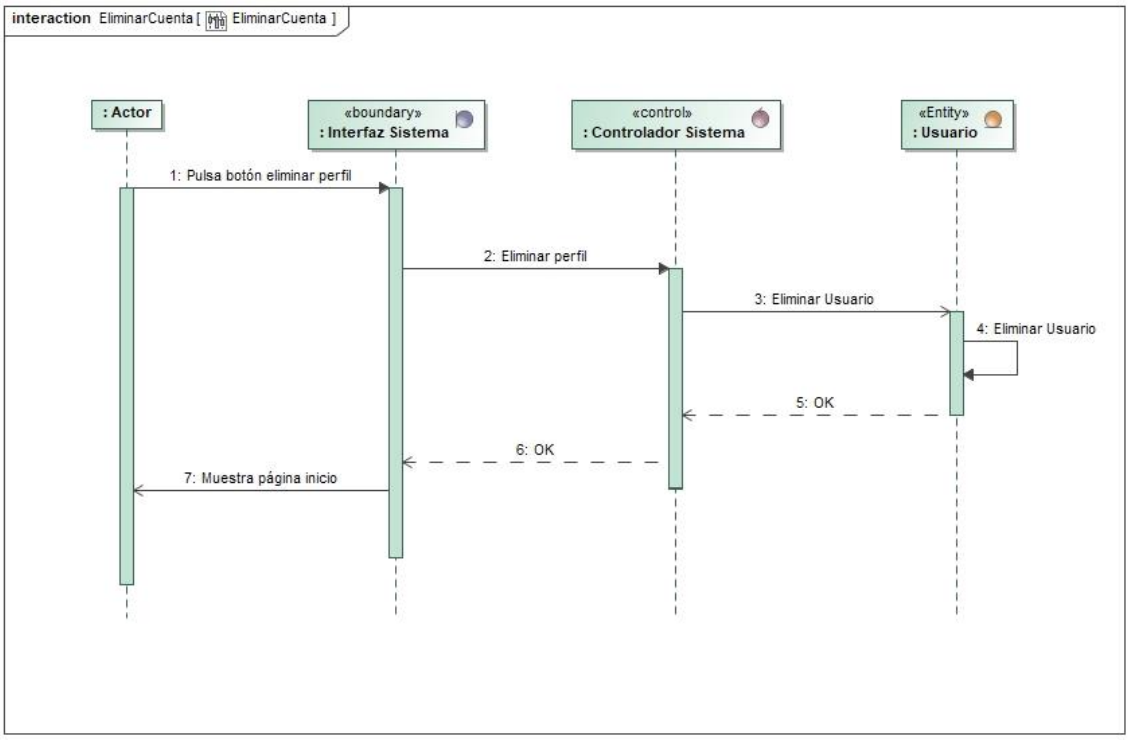

Figura 4. 15 : Eliminar Perfil

4.4 Diseño

Conjunto de planes y decisiones para definir un producto con los suficientes detalles como para permitir su realización física de acuerdo a unos requisitos.

Arquitectura del sistema

La arquitectura se basa en un diseño basado en componentes. Este tipo de diseño consiste en descomponer el sistema en componentes funcionales que implementen una interfaz de comunicación entre ellos. Este tipo de diseño posibilita la obtención de una serie de ventajas como la reutilización de los componentes, conseguir una escalabilidad del sistema, una abstracción entre los componentes gracias a esa interfaz de comunicación, entre otras. En la pasada Figura 4.1, se pueden apreciar estos componentes.

Comunicación

La comunicación entre los componentes que hacen uso de Apache Kafka tanto aquellos que utilizan el protocolo HTTP, usan JSON (Java Script Object Notation) como formato de los datos. Para la transmisión de los bytes de la imagen entre el cliente y servidor leshan, se utiliza TLV (type-length-value, tipo-longitud-valor).

JSON es un estándar basado en texto plano para el intercambio de información. Una de las ventajas destacables es que resulta ser un estándar independiente del lenguaje de programación, por lo que permite que los servicios o componentes que comparten información, puedan estar escritos en lenguajes diferentes. (Cadenas, 2016)

JSON está estructurado en:

- Una colección de pares nombre/valor, lo que también se conoce como objeto en algunos lenguajes.
- Una lista de valores.

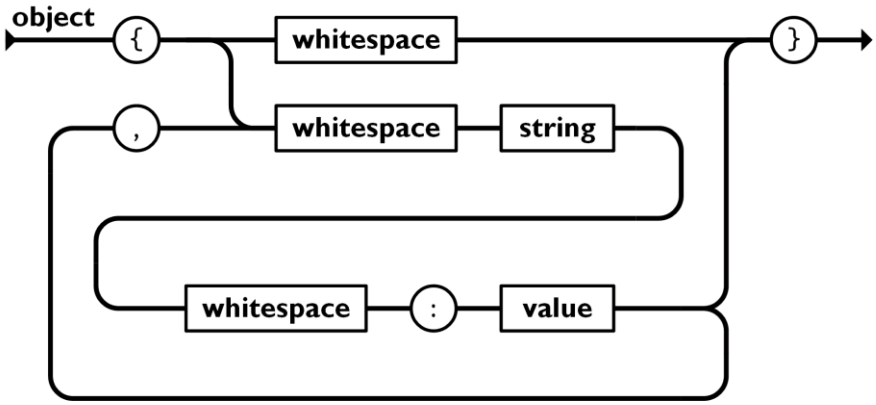

Figura 4. 16 : Objeto JSON

TLV o valores de longitud tipo, en castellano, es un formato de representación de información que permite la opcionalidad de información y que ésta sea variable. A continuación se muestra la respuesta que emite el cliente cuando envía los datos de la imagen:

```
{"ep": "urn:oma:lwm2m:x:39574", "res": "/39574/0/27366", "val": {"id": 27366, "value": "…"}}
```

En los puntos suspensivos estaría el string correspondiente a los bytes de una parte de la imagen.

Diagrama Entidad-Relación

En esta ocasión, tenemos dos bases de datos mongoDB, en una de ellas almacenamos las imágenes resultantes del análisis, y en la otra a los usuarios de la aplicación web. Cada base de datos está compuesta de una sola entidad, imagen y usuario respectivamente. Por lo tanto solo consta de dos entidades que no comparten relación entre ellas, ya que cualquier usuario que se registra, tiene acceso para ver las imágenes que se han analizado.

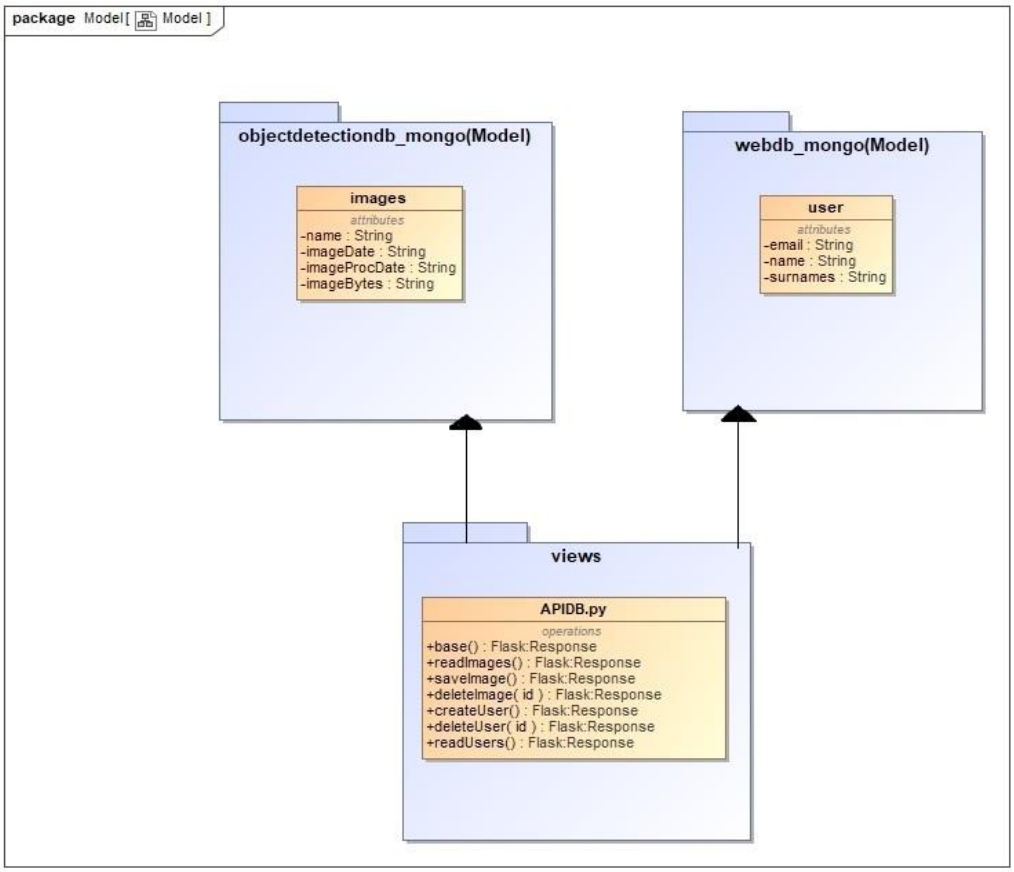

Figura 4. 17 : Diagrama de Base de Datos y API

Algoritmo Detección de Objetos

Como se ha mencionado con anterioridad, el algoritmo elegido para realizar la detección de objetos ha sido *YOLO.*

YOLO es un algoritmo de código abierto cuya finalidad es detectar objetos en tiempo real. Para realizar esta detección, el algoritmo divide la imagen en una cuadrícula NxN, tal y como se puede apreciar en la primera imagen de la Figura 4.18. Para cada una de las celdas predice posibles regiones de identificación y su certidumbre (segunda imagen). Tras esto, elimina aquellas regiones cuyo nivel de certidumbre es inferior a un límite dado. A las regiones restantes se le aplica un paso para eliminar los objetos que hayan podido ser detectados por duplicado, y el resultado se puede apreciar en la última imagen de la figura.

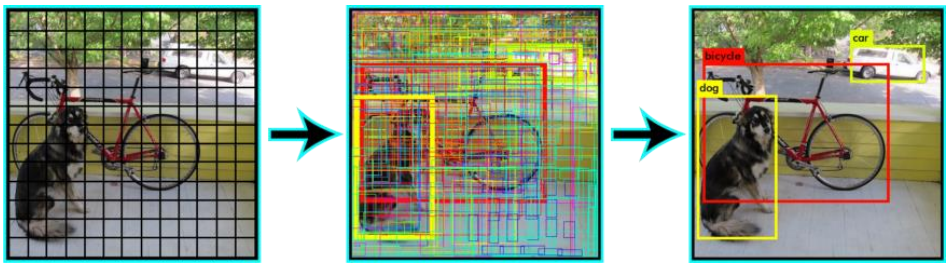

Figura 4. 18 : Procedimiento de Algoritmo YOLO

Aplicación Web

Como se ha mencionado, la función de la aplicación web es la de visualizar las imágenes resultantes del análisis, por lo que el diseño de ésta es simple. Para el inicio de sesión, se hace uso de Firebase que proporciona autenticación con Google. Las imágenes se listarán en una tabla y al pulsar sobre el botón se mostrará el contenido. El usuario podrá en todo momento eliminar su cuenta desde la pestaña *Profile*.

En las siguientes figuras, se puede apreciar unos bocetos del diseño de la aplicación web

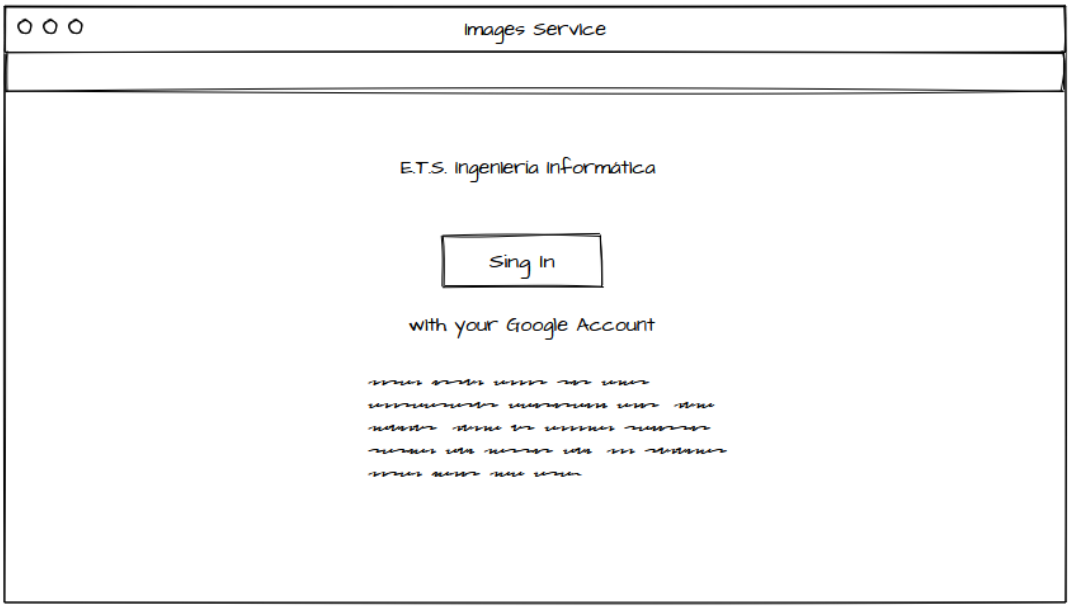

Figura 4. 19: Vista Inicio

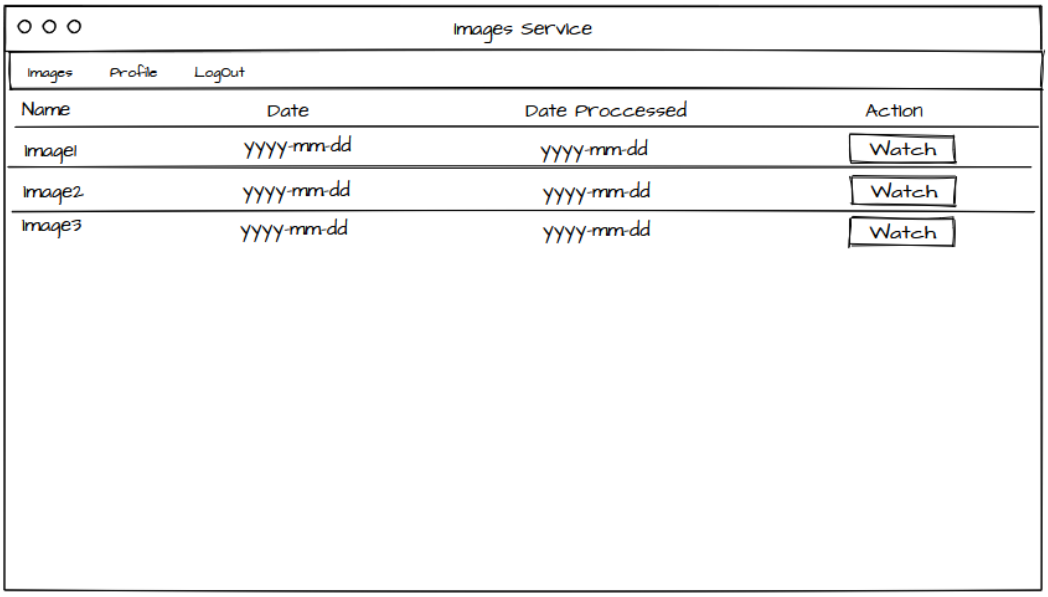

Figura 4. 20: Vista Principal

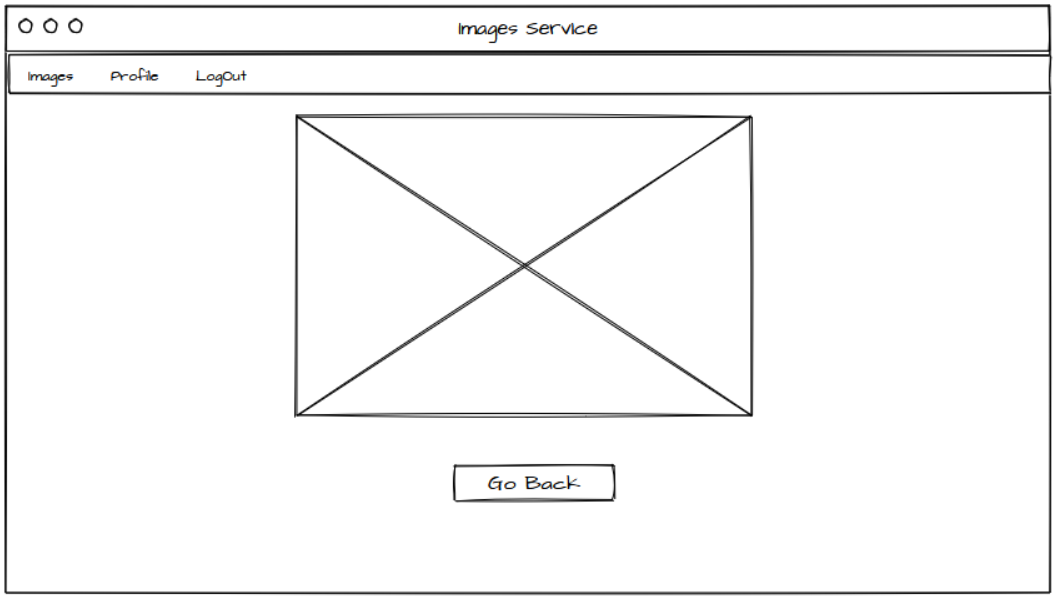

Figura 4. 21: Vista de Imagen

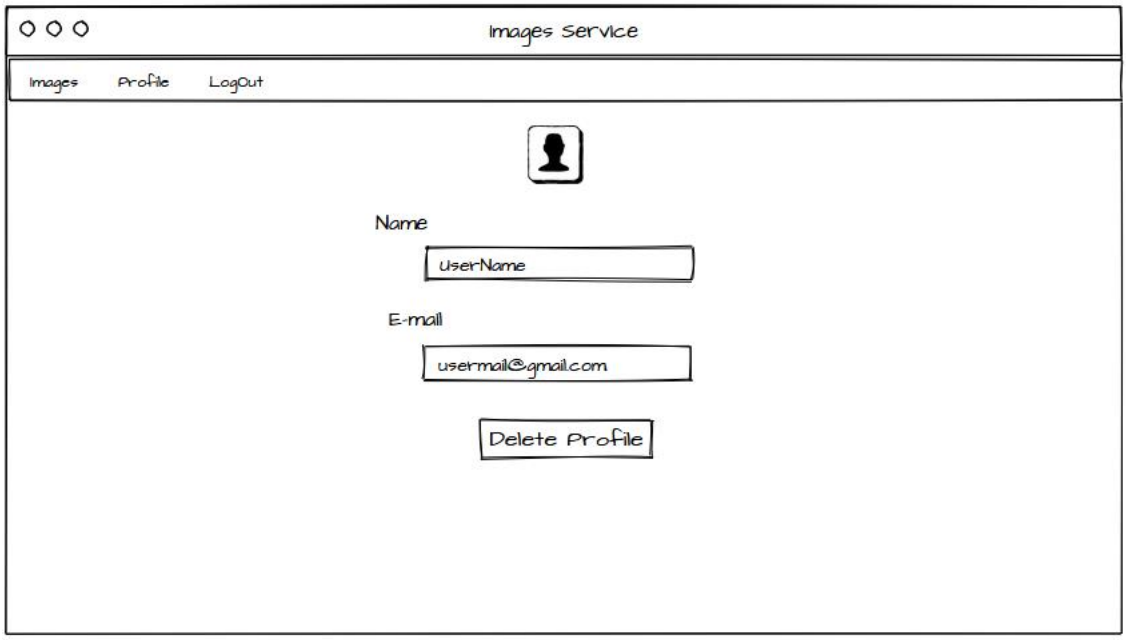

Figura 4. 22: Vista de Perfil

4.5 Implementación

La implementación es otra etapa del desarrollo de software. Aquí es donde se materializa toda la documentación generada con anterioridad.

En este apartado se procede a la descripción de la implementación de los módulos desarrollados en este proyecto ayudados por su correspondiente diagrama.

Cliente Leshan

El módulo ***IoT_LeshanClient,*** tiene como objetivo transmitir los datos sobre las imágenes. Para ello procesa las peticiones que le envía el servidor Leshan.

Este módulo ha sido desarrollado en Java 9 y hace uso de la librería de leshan v1.1.0.

En la Figura 4.23 se puede apreciar el diagrama de clases de este módulo.

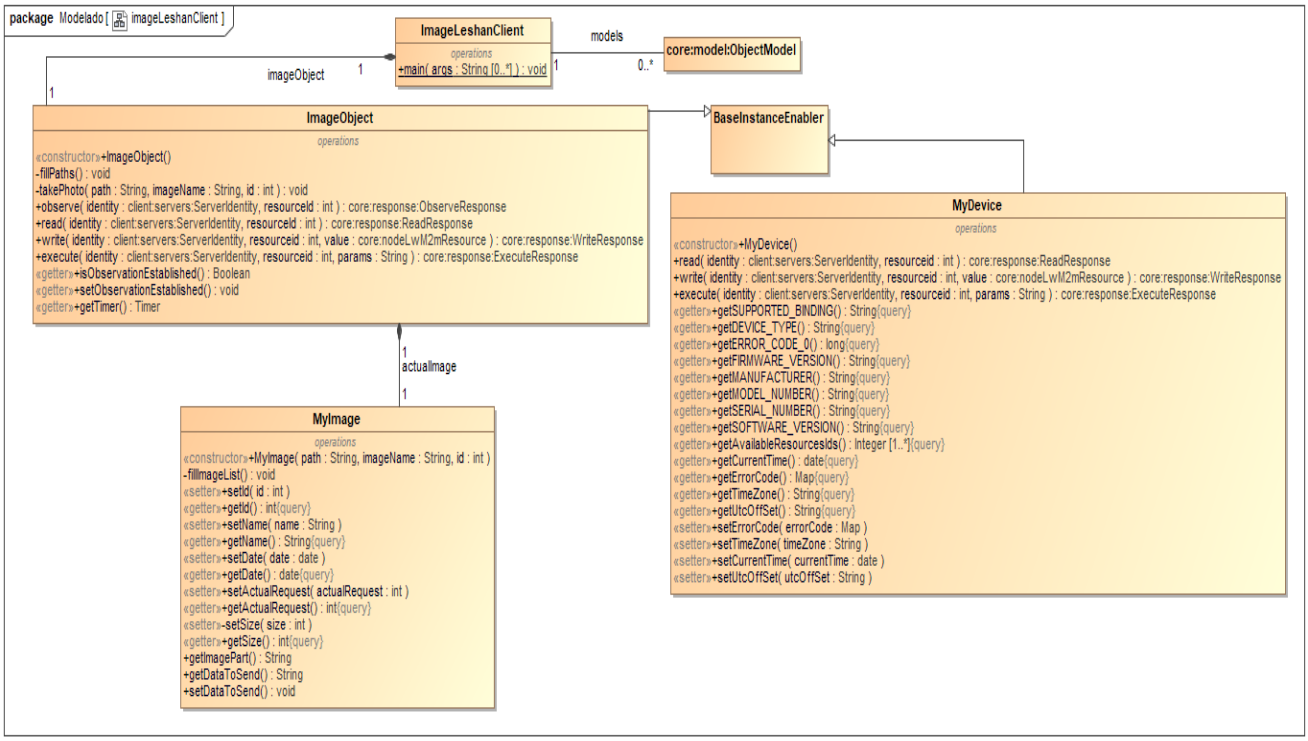

Figura 4. 23: Diagrama de Clases Cliente Leshan

La clase principal, ***ImageLeshanClient*** es la encargada de inicializar el cliente Leshan, cargando toda la información de conexión y los objetos que lo conforman.

El primer objeto, *Device,* corresponde a la clase ***MyDevice***, este objeto es obligatorio para todo cliente, pues así lo indica el protocolo *LWM2M*. Este objeto

contiene información relevante del dispositivo. Alguna información como *ERROR_CODE* es obligatoria que la posea, mientras el resto es opcional. En este proyecto se ha decidido implementar cierta información opcional, para que nuestro dispositivo sea más completo. Esta clase posee los métodos getters y setters correspondientes a los recursos del objeto *Device* que pueden ser leídos, modificados u observados.

El segundo objeto, *Image Data*, corresponde con la clase ***ImageObject***, este objeto es el encargado de contener toda la funcionalidad necesaria para que se pueda transmitir las imágenes. La clase ***ImageObject,*** implementa la funcionalidad del objeto correspondiente al tratamiento de las peticiones. Esta clase, está compuesta por un objeto de la clase ***MyImage***, que es la que contiene la información de la imagen que actualmente está cargada en el sistema. La clase ***ImageObject***, hace uso de los métodos getters y setters implementados en ***MyImage*** para transmitir la información correspondiente del recurso solicitado.

De este módulo destacamos las siguientes funciones:

- La función ***observe(…)***, es la encargada de establecer una observación sobre un determinado recurso, cuando recibe este tipo de peticiones. Esta función es bastante importante en nuestro sistema, pues es la encargada de transmitir los bytes de la imagen. Para ello hace uso de un ***Timer*** que cada 2 segundos produce un evento *Notify* y de los métodos ***setDataToSend()***, esta función hace uso de ***getImagepart()***, *y* ***getDataToSend()*** .

- La función ***execute(..)***, es la encargada de procesar las peticiones *EXECUTE*. En este objeto, se utiliza para generar un nuevo objeto ***MyImage***, que contiene la información de otra imagen del sistema. Este método simula la toma de una nueva fotografía.

- La función ***read(…),*** se encarga de procesar las peticiones *READ,* para ello hace uso de los métodos getters y setters de la clase ***MyImage***.

- La función ***write(..)***, se encarga de procesar las peticiones *WRITE,* estas peticiones se utilizan para modificar el valor de un determinado recurso. En el caso del objeto *Image Data*, no se ha implementado ningún tipo de funcionalidad para este tipo de peticiones. En cambio, para el objeto ***Device***, si se ha implementado la modificación de un par de recursos.

Estas funciones que se han destacado anteriormente, son de obligada implementación, puesto que son las encargadas de procesas las peticiones *OBSERVE, READ, WRITE, EXECUTE*.

Aplicación de detección de objetos

El módulo ***IoT_Object_Detection_App***, tiene como objetivo realizar peticiones a un cliente Leshan para obtener los datos de una imagen. Para ello hace uso del módulo *iotshadowapplications, iotconnector* y *kafka.*

Este módulo ha sido desarrollado en Python 3.4, haciendo uso de las siguientes librerías:

- kafka-python v2.0.1
- matplotlib v3.2.2
- numpy v1.5.0
- opencv-Python v4.2.0.34
- pybase v1.0.1
- python-dateutil v2.8.1
- request v2.21.0

En la Figura 4.24 se puede ver el diagrama de clases de este módulo,

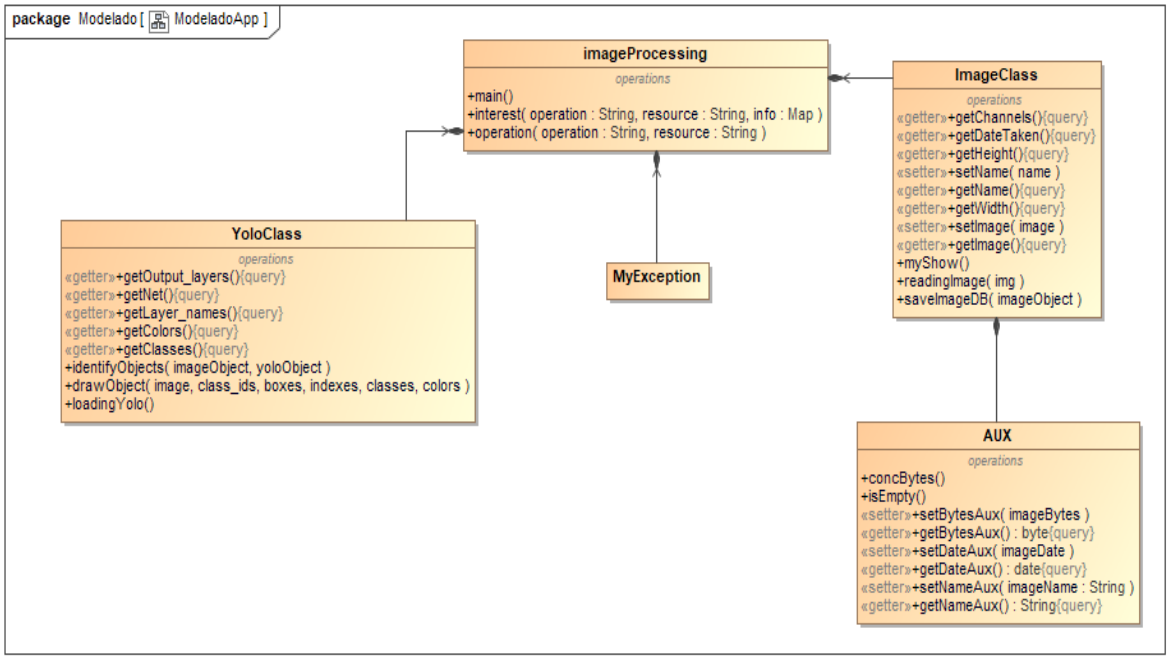

Figura 4. 24: Diagrama de Clases de Aplicación Detección de Objetos

La clase principal, ***imageProcessing.py*** que contiene la función ***interest(…)*** que usada para mostrar interés sobre un determinado recurso. La función ***operation(…)*** que es utilizada para realizar una acción sobre un determinado recurso, como se ha mencionado con anterioridad, esta acción puede ser *READ, WRITE, OBSERVE, CANCEL_OBSERVATION, EXECTUTE.* La función ***main()*** contiene la funcionalidad principal de la aplicación, es la encargada de crear el objeto de la clase *YoloClass.py*, y de crear el objeto de la clase *ImageClass.py* .

Una clase ***MyException.py***, que es utilizada para lanzar excepciones cuando se produce algún error.

La clase ***YoloClass.py*** contiene la implementación del algoritmo *YOLO* para realizar el análisis de la imagen. Para ello debe cargar el archivo de configuración ***yolov3.cfg*** , el modelo entrenado ***yolov3.weights*** y el archivo que contiene los nombres para identificar el objeto ***coco.names*** . La función ***loadingYolo()*** es llamada desde la función ***main()*** de ***imageProcessing.py*** para cargar todos los archivos anteriores y asignar valor a las respectivas variables.

La clase ***ImageClass.py*** contiene todos los atributos necesarios para que el algoritmo YOLO pueda ejecutarse correctamente. Hace uso de una clase auxiliar, ***AUX.py*** que es utilizada para cargar los datos recibidos de la imagen.

Aplicación web

Esta aplicación web, ***Images_Web***, tiene como objetivo visualizar las imágenes que se han analizado con anterioridad. Para ello hace uso de la *API REST* que se puede apreciar en la Figura 4.1 para la comunicación con la base de datos.

Este módulo ha sido desarrollado en Python 3.7.3 haciendo uso de las siguientes librerías y frameworks:

- Flask v1.1.2
- Jinja2 v2.11.2
- Unumpy v1.19.0
- opencv-python v4.2.0.34
- pymongo v3.10.1
- requests v2.21.0
- Werkzeug v1.0.1

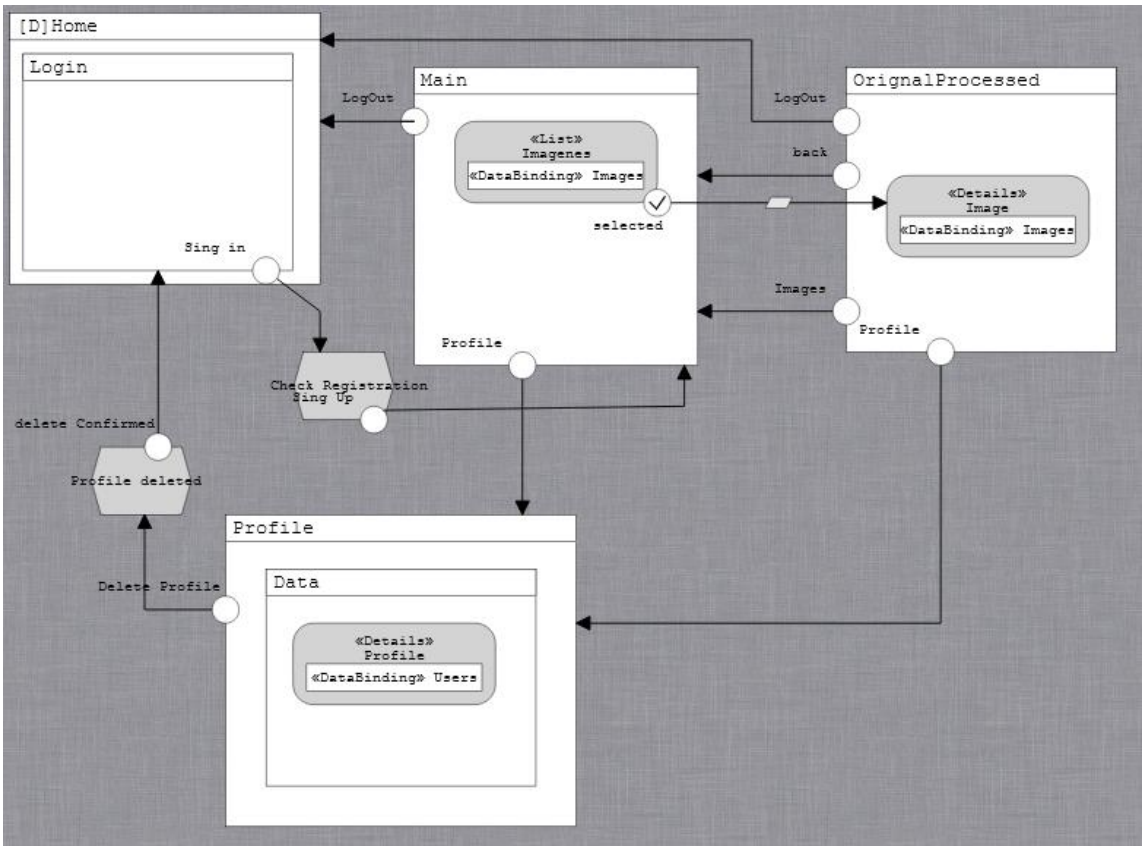

Figura 4. 25: Modelo de Aplicación Web

Esta aplicación web tiene una vista inicial, en la cual deberemos iniciar sesión haciendo uso de nuestra cuenta de *Google*, una vez se ha iniciado sesión, la aplicación comprueba si este usuario ya ha sido registrado, en caso negativo, realiza un registro del usuario. Una vez ingresados en la aplicación, tenemos una vista principal donde se nos muestra una lista de todas las imágenes que hay almacenadas en ese momento en la base de datos. También disponemos de un panel de navegación en el cual podemos cerrar sesión o irnos a la vista ***Profile***. En esta vista, podremos ver la información recopilada de nuestro usuario por parte de la aplicación; tendremos la posibilidad de eliminar nuestro perfil. Volviendo a la vista principal, a través del panel de navegación, podemos seleccionar qué imagen visualizar, una vez seleccionada se nos mostrara en una nueva vista la imagen procesada.

4.6 Evaluación

Para poder realizar una evaluación sobre el comportamiento del sistema bajo la carga de trabajo que supone realizar el análisis de una imagen recibida, se pueden ejecutar una serie de comprobaciones y en base a su resultado, considerar si es posible, añadir modificaciones que permitan adecuar el sistema su a propósito.

A continuación se listan una serie de pruebas que se pueden aplicar sobre el sistema:

- Tiempo medio de envío de una imagen.
- Tiempo medio de procesamiento de una imagen
- Tiempo desde que se realiza una petición al recurso por parte de la aplicación hasta que recibe el primer dato.
- Tiempo desde la muestra de interés por parte de la aplicación hasta que recibe la respuesta del recurso.
- Número de peticiones perdidas.
- Número de respuestas perdidas.

Todas estas pruebas, se pueden realizar con más de una aplicación haciendo peticiones ,o bien, a un mismo cliente, o a clientes distintos.

A continuación, se muestra el resultado de una de las pruebas realizadas, en las que se ha calculado el tiempo medio de envío de una imagen:

Tiempo de la sesión *(hh:mm:ss)* → *7:38:30*

Total Imágenes Transmitidas → 100

Tiempo medio de Transmisión *(hh:mm:ss)* → *00:03:43*

Total de datos de imagen enviados → *7746 KB.*

Menor Tiempo de Transmisión *(hh:mm:ss)* → *00:03:32* (corresponde a la imagen *animals.jpg*, con un tamaño de 8,1 KB).

Mayor Tiempo de Transmisión *(hh:mm:ss)* → *00:04:26* (corresponde a la imagen *cyclist.jpg*, con un tamaño de 164 KB).

Imagen	Hora de la Imagen (hh:mm:ss)	Hora de procesamiento (hh:mm:ss)	Tamaño Imagen (KB)	Tiempo total (hh:mm:ss)
animals.jpg	12:40:18	12:43:51	8,1	0:03:33
laptop.jpg	12:44:43	12:48:16	22,6	0:03:33
fruits.jpg	12:49:08	12:52:41	81,6	0:03:33
person.jpg	12:53:33	12:57:06	111	0:03:33
cyclist.jpg	12:57:58	13:02:24	164	0:04:26
animals.jpg	13:03:16	13:06:49	8,1	0:03:33
laptop.jpg	13:07:41	13:11:14	22,6	0:03:33
fruits.jpg	13:12:07	13:15:39	81,6	0:03:32
person.jpg	13:16:32	13:20:04	111	0:03:32
cyclist.jpg	13:20:57	13:25:22	164	0:04:25
animals.jpg	13:26:15	13:29:47	8,1	0:03:32
laptop.jpg	13:30:40	13:34:12	22,6	0:03:32
fruits.jpg	13:35:05	13:38:37	81,6	0:03:32
person.jpg	13:39:30	13:43:02	111	0:03:32
cyclist.jpg	13:43:55	13:48:20	164	0:04:25
animals.jpg	13:49:13	13:52:46	8,1	0:03:33
laptop.jpg	13:53:38	13:57:10	22,6	0:03:32
fruits.jpg	13:58:03	14:01:35	81,6	0:03:32
person.jpg	14:02:28	14:06:00	111	0:03:32
cyclist.jpg	14:06:53	14:11:19	164	0:04:26
animals.jpg	14:12:11	14:15:44	8,1	0:03:33
laptop.jpg	14:16:36	14:20:09	22,6	0:03:33
fruits.jpg	14:21:01	14:24:34	81,6	0:03:33
person.jpg	14:25:26	14:28:59	111	0:03:33

cyclist.jpg	14:29:51	14:34:17	164	0:04:26
animals.jpg	14:35:09	14:38:42	8,1	0:03:33
laptop.jpg	14:39:34	14:43:07	22,6	0:03:33
fruits.jpg	14:43:59	14:47:32	81,6	0:03:33
person.jpg	14:48:24	14:51:57	111	0:03:33
cyclist.jpg	14:52:49	14:57:15	164	0:04:26
animals.jpg	14:58:07	15:01:40	8,1	0:03:33
laptop.jpg	15:02:32	15:06:05	22,6	0:03:33
fruits.jpg	15:06:57	15:10:30	81,6	0:03:33
person.jpg	15:11:22	15:14:55	111	0:03:33
cyclist.jpg	15:15:47	15:20:13	164	0:04:26
animals.jpg	15:21:05	15:24:38	8,1	0:03:33
laptop.jpg	15:25:30	15:29:03	22,6	0:03:33
fruits.jpg	15:29:55	15:33:28	81,6	0:03:33
person.jpg	15:34:20	15:37:53	111	0:03:33
cyclist.jpg	15:38:45	15:43:11	164	0:04:26
animals.jpg	15:44:03	15:47:36	8,1	0:03:33
laptop.jpg	15:48:28	15:52:01	22,6	0:03:33
fruits.jpg	15:52:53	15:56:26	81,6	0:03:33
person.jpg	15:57:18	16:00:51	111	0:03:33
cyclist.jpg	16:01:43	16:06:09	164	0:04:26
animals.jpg	16:07:01	16:10:34	8,1	0:03:33
laptop.jpg	16:11:26	16:14:59	22,6	0:03:33
fruits.jpg	16:15:51	16:19:24	81,6	0:03:33
person.jpg	16:20:16	16:23:49	111	0:03:33
cyclist.jpg	16:24:41	16:29:07	164	0:04:26
animals.jpg	16:29:59	16:33:32	8,1	0:03:33
laptop.jpg	16:34:24	16:37:57	22,6	0:03:33
fruits.jpg	16:38:49	16:42:22	81,6	0:03:33
person.jpg	16:43:14	16:46:47	111	0:03:33
cyclist.jpg	16:47:39	16:52:05	164	0:04:26
animals.jpg	16:52:57	16:56:30	8,1	0:03:33
laptop.jpg	16:57:22	17:00:55	22,6	0:03:33
fruits.jpg	17:01:47	17:05:20	81,6	0:03:33
person.jpg	17:06:12	17:09:45	111	0:03:33
cyclist.jpg	17:10:37	17:15:03	164	0:04:26

animals.jpg	17:15:55	17:19:28	8,1	0:03:33
laptop.jpg	17:20:20	17:23:53	22,6	0:03:33
fruits.jpg	17:24:45	17:28:18	81,6	0:03:33
person.jpg	17:29:10	17:32:43	111	0:03:33
cyclist.jpg	17:33:35	17:38:01	164	0:04:26
nimals.jpg	17:38:53	17:42:26	8,1	0:03:33
laptop.jpg	17:43:18	17:46:51	22,6	0:03:33
fruits.jpg	17:47:43	17:51:16	81,6	0:03:33
person.jpg	17:52:08	17:55:41	111	0:03:33
cyclist.jpg	17:56:33	18:00:59	164	0:04:26
animals.jpg	18:01:52	18:05:24	8,1	0:03:32
laptop.jpg	18:06:17	18:09:49	22,6	0:03:32
fruits.jpg	18:10:42	18:14:14	81,6	0:03:32
person.jpg	18:15:07	18:18:39	111	0:03:32
cyclist.jpg	18:19:32	18:23:57	164	0:04:25
animals.jpg	18:24:50	18:28:22	8,1	0:03:32
laptop.jpg	18:29:15	18:32:47	22,6	0:03:32
fruits.jpg	18:33:40	18:37:12	81,6	0:03:32
person.jpg	18:38:05	18:41:37	111	0:03:32
cyclist.jpg	18:42:30	18:46:55	164	0:04:25
animals.jpg	18:47:48	18:51:20	8,1	0:03:32
laptop.jpg	18:52:13	18:55:45	22,6	0:03:32
fruits.jpg	18:56:38	19:00:10	81,6	0:03:32
person.jpg	19:01:03	19:04:35	111	0:03:32
cyclist.jpg	19:05:28	19:09:54	164	0:04:26
animals.jpg	19:10:46	19:14:19	8,1	0:03:33
laptop.jpg	19:15:11	19:18:44	22,6	0:03:33
fruits.jpg	19:19:36	19:23:08	81,6	0:03:32
person.jpg	19:24:01	19:27:33	111	0:03:32
cyclist.jpg	19:28:26	19:32:52	164	0:04:26
animals.jpg	19:33:44	19:37:16	8,1	0:03:32
laptop.jpg	19:38:09	19:41:41	22,6	0:03:32
fruits.jpg	19:42:34	19:46:07	81,6	0:03:33
person.jpg	19:46:59	19:50:31	111	0:03:32
cyclist.jpg	19:51:24	19:55:50	164	0:04:26
animals.jpg	19:56:42	20:00:15	8,1	0:03:33

laptop.jpg	20:01:07	20:04:40	22,6	0:03:33
fruits.jpg	20:05:32	20:09:05	81,6	0:03:33
person.jpg	20:09:57	20:13:30	111	0:03:33
cyclist.jpg	20:14:22	20:18:48	164	0:04:26

4.7 Iteraciones

Este proyecto se ha llevado a cabo mediante cinco iteraciones que se describen a continuación.

Iteración 0

En esta iteración no se ha llevado a cabo ninguna implementación de los requisitos enumerados con anterioridad. Esta iteración ha servido para estudiar el funcionamiento de la arquitectura que se procede a evaluar con este proyecto, además de comprender el protocolo de comunicación *LWM2M* que se utiliza para la comunicación entre el cliente a desarrollar y el servidor de demostración que proporciona el proyecto Leshan. También se ha procedido a la elección del algoritmo de detección de objetos en imágenes, y a la creación de pequeñas aplicaciones que en base a ellas se ha procedido a desarrollar módulos para el proyecto. Esta iteración ha sido una toma de contacto con las tecnologías que se iban a utilizar para la elaboración del trabajo.

Iteración 1

En esta primera iteración se ha procedido al análisis, especificación, diseño e implementación del módulo del sistema embebido.

Iteración 2

En esta segunda iteración se ha procedido al análisis, especificación, diseño e implementación del módulo de análisis de imágenes, además se produjo el desarrollo de la API REST que permite el almacenamiento de las imágenes en la base de datos Mongo.

Iteración 3

En esta tercera iteración se ha procedido al análisis, especificación, diseño e implementación del módulo de la aplicación web, además se implementó el resto de la funcionalidad de la API REST correspondiente a la comunicación con este módulo.

Iteración 4

En esta última iteración se ha procedido a la integración de los módulos desarrollados con la arquitectura, a la resolución de incompatibilidades y errores de comunicación entre la arquitectura y los módulos.

5

Conclusiones

La elaboración de este proyecto ha supuesto el desarrollo de un sistema capaz de transmitir imágenes a una arquitectura edge tolerante a fallos hardware y software, en la que se lleva a cabo un procesamiento de los datos y que permite acceder al resultado a través de una aplicación web.

Como resultado final, podemos concluir que esta arquitectura es capaz de soportar la carga de trabajo que el procesamiento de imágenes requiere, y como nuestro objetivo principal consiste en la evaluación de dicha arquitectura, podemos concluir que el balance de trabajo de la arquitectura es positivo.

Al poseer todo el sistema una estructura modular, es posible que un futuro se puedan integrar nuevos módulos auxiliares para el procesamiento de imágenes, como la creación de una aplicación más completa o desarrollar otros módulos que hagan uso de este procesamiento.

Por otra parte, la utilización de la metodología de desarrollo adoptada ha supuesto una ventaja, ya que en las distintas reuniones con los tutores se han producido cambios o modificaciones en los requisitos que han permitido abordarlos en las distintas iteraciones del proyecto que aquí se presenta.

En cuanto a la experiencia personal, ha supuesto todo un reto desde el primer momento. He tenido que familiarizarme con un tipo de arquitectura desconocida para mi inicialmente, he tenido que aprender y comprender por primera vez un protocolo de comunicación de dispositivos IoT, y además, hacer uso de un algoritmo de visión artificial. En definitiva, ha sido un proyecto muy interesante, pues me ha enriquecido como futuro ingeniero, ya que considero que he adquirido y profundizado en

conocimientos y nuevas tecnologías que han supuesto todo un reto, además de otorgarme interés sobre los dispositivos IoT, arquitecturas edge e inteligencia artificial.

Referencias

A. (s.f.). Obtenido de https://timber.io/blog/hello-world-in-kafka-using-python/

A., E. (12 de Mayo de 2018). *Medium*. Recuperado el 21 de Enero de 2021, de Medium: https://medium.com/@enriqueav/detecci%C3%B3n-de-objetos-con-yolo-implementaciones-y-como-usarlas-c73ca2489246

Alves, S. (04 de Septiembre de 2019). *venturus*. Recuperado el 21 de Enero de 2021, de venturus: https://www.venturus.org.br/en/edge-computing-and-its-applications-in-industry-4-0/

Amazon Web Service. (s.f.). *Amazon Web Service*. Recuperado el 13 de Enero de 2021, de https://aws.amazon.com/es/docker/

Apache Kafka. (s.f.). *kafka*. Recuperado el 14 de Enero de 2021, de kafka: https://kafka.apache.org/

Apache Kafka. (s.f.). *kafka.apache.org.* Recuperado el 26 de Julio de 2020, de kafka.apache.org: https://kafka.apache.org/documentation/#producerapi

Blogs Itpro. (22 de Agosto de 2016). *Blogs Itpro*. Recuperado el 14 de Enero de 2021, de Blogs Itpro: https://blogs.itpro.es/eduardocloud/2016/08/22/visual-studio-code-que-es-y-que-no-es/

Bootstrap. (s.f.). *Bootstrap*. Recuperado el 15 de Septiembre de 2020, de Bootstrap: https://getbootstrap.com/docs/3.3/components/

Cadenas, V. G. (17 de Julio de 2016). *Medium*. Recuperado el 20 de Enero de 2021, de Medium: https://medium.com/@victor.garibayy/qu%C3%A9-es-y-para-qu%C3%A9-sirve-json-be05fe02e67d

CALENDAMAIA. (10 de Enero de 2014). *GENBETA*. Recuperado el 14 de Enero de 2021, de GENBETA: https://www.genbeta.com/desarrollo/eclipse-ide

Camilo, D. (24 de Junio de 2020). *HUMMINGBIRDS.* Recuperado el 1 de Febrero de 2021, de HUMMINGBIRDS: http://hummingbirds.ai/iot-and-edge-computing-for-computer-vision/

Cendón, B. (16 de Enero de 2017). *BRUNO CENDÓN.* Recuperado el 25 de Enero de 2021, de BRUNO CENDÓN: http://www.bcendon.com/el-origen-del-iot/

Combs, V. (30 de Octubre de 2019). *TechRepublic.* Recuperado el 1 de Febrero de 2021, de TechRepublic: https://www.techrepublic.com/article/intel-and-ge-healthcares-x-ray-machine-uses-embedded-ai-to-prioritize-scans/

Cualquiera, L. (6 de Febrero de 2020). *Medium*. Recuperado el 16 de Noviembre de 2020, de Medium: https://medium.com/@luiscualquiera/c%C3%B3mo-gestionar-los-commit-en-kafka-4a9ff18763c3

Cuervo, V. (9 de Marzo de 2016). *Línea de Código*. Recuperado el 14 de Enero de 2021, de http://lineadecodigo.com/python/hola-mundo-pymongo/

Developer Mozilla. (4 de Junio de 2020). *Developer Mozilla*. Recuperado el 17 de Septiembre de 2020, de Developer Mozilla: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/send

Docker. (s.f.). *Docker*. Recuperado el 13 de Enero de 2021, de https://www.docker.com/resources/what-container

Docker. (s.f.). *Docker.* Recuperado el 29 de Junio de 2020, de Docker: https://docs.docker.com/engine/reference/commandline

Docker. (s.f.). *Docker*. Recuperado el 25 de Agosto de 2020, de Docker: https://docs.docker.com/compose/compose-file/

Docker. (s.f.). *Docker Docs*. Recuperado el 13 de Enero de 2021, de Docker Docs: https://docs.docker.com/compose/

Docker. (s.f.). *Docker Docs*. Recuperado el 7 de Noviembre de 2020, de Docker Docs: https://docs.docker.com/config/containers/multi-service_container/

Docker Docs. (s.f.). *Docker Docs*. Recuperado el 5 de Noviembre de 2020, de Docker Docs: https://docs.docker.com/engine/swarm/how-swarm-mode-works/swarm-task-states/

Docker Docs. (s.f.). *Docker Docs*. Recuperado el 5 de Noviembre de 2020, de Docker Docs: https://docs.docker.com/reference/

Docker Docs. (s.f.). *Docker Docs*. Recuperado el 2 de Diciembre de 2020, de Docker Docs: https://docs.docker.com/engine/reference/commandline/service_create/

Dorpe, S. V. (13 de Agosto de 2018). *Towards Data Science.* Recuperado el 26 de Julio de 2020, de Towards Data Science: https://towardsdatascience.com/kafka-python-explained-in-10-lines-of-code-800e3e07dad1

Eclipse . (s.f.). *Eclipse* . Recuperado el 13 de Enero de 2021, de https://www.eclipse.org/leshan/

Eclipse Leshan. (20 de Marzo de 2020). *GitHub.* Recuperado el 3 de Julio de 2020, de GitHub: https://github.com/eclipse/leshan/wiki

Eclipse Leshan. (16 de Abril de 2020). *GitHub.* Recuperado el 30 de Julio de 2020, de GitHub: https://github.com/eclipse/leshan/blob/master/README.md

Eclipse Leshan. (15 de Abril de 2020). *GitHub.* Recuperado el 30 de Julio de 2020, de GitHub: https://github.com/eclipse/leshan/wiki/Getting-Started-:-Client

EcuRed. (6 de Julio de 2019). *EcuRed.* Recuperado el 14 de Enero de 2021, de EcuRed: https://www.ecured.cu/Metodologias_de_desarrollo_de_Software#Proceso_Unificado_de_Desarrollo_.28RUP.29

EcuRed. (s.f.). *EcuRed*. Recuperado el 14 de Enero de 2021, de EcuRed: https://www.ecured.cu/MagicDraw

Firebase. (3 de Diciembre de 2019). *Firebase*. Recuperado el 16 de Septiembre de 2020, de Firebase: https://firebase.google.com/docs/auth/web/start?authuser=0

Firebase. (2 de Julio de 2020). *Firebase*. Recuperado el 16 de Septiembre de 2020, de Firebase: https://firebase.google.com/docs/auth/web/google-signin?authuser=0

Firebase. (18 de Junio de 2020). *Firebase*. Recuperado el 16 de Septiembre de 2020, de Firebase: https://firebase.google.com/docs/auth/web/manage-users?hl=es-419

Galindo, J. C. (17 de Noviembre de 2020). *Muy Interesante*. Recuperado el 22 de Enero de 2021, de Muy Interesante: https://www.muyinteresante.es/tecnologia/inteligencia-artificial/articulo/5g-maritime-un-proyecto-portuario-pionero-que-utiliza-ia-y-5g-931605557957

GE Healthcare. (s.f.). *GE Healthcare.* Recuperado el 1 de Febrero de 2021, de GE Healthcare: https://www.gehealthcare.es/products/radiography/mobile-xray-systems/critical-care-suite-on-optima-xr240amx

Gherking. (s.f.). *Gherking*. Recuperado el 30 de Agosto de 2020, de Gherking: https://cucumber.io/docs/gherkin/reference/

GitHub. (s.f.). *GitHub*. Recuperado el 17 de Septiembre de 2020, de GitHub: https://gist.github.com/KentaYamada/2eed4af1f6b2adac5cc7c9063acf8720

Gómez, M. A. (s.f.). *Software Crafters*. Recuperado el 13 de Enero de 2021, de Software Crafters: https://softwarecrafters.io/devops/cluster-docker-swarm-digital-ocean

gradiant. (12 de Abril de 2018). *gradiant*. Recuperado el 21 de Enero de 2021, de gradiant: https://www.gradiant.org/blog/edge-fog-computing-cloud/

Hamdaboy. (39 de Junio de 2016). Recuperado el 3 de Julio de 2020, de https://www.slideshare.net/Hamdamboy/the-constrained-application-protocol-coap-63506082

Hernández, R. (1 de Junio de 2016). *GENBETA.* Recuperado el 17 de Enero de 2021, de GENBETA: https://www.genbeta.com/desarrollo/bdd-cucumber-y-gherkin-desarrollo-dirigido-por-comportamiento

Ionos. (5 de Diciembre de 2019). *Ionos*. Obtenido de Ionos: https://www.ionos.es/digitalguide/servidores/know-how/edge-computing/

IONOS. (9 de Julio de 2019). *IONOS*. Recuperado el 29 de 06 de 2020, de https://www.ionos.es/digitalguide/servidores/know-how/docker-compose-y-swarm-gestion-multicontenedor/

IONOS. (9 de Julio de 2019). *IONOS.* Recuperado el 29 de Junio de 2020, de IONOS: https://www.ionos.es/digitalguide/servidores/know-how/docker-compose-y-swarm-gestion-multicontenedor/

Jardinet, T. (30 de Enero de 2020). *DZone*. Recuperado el 21 de Enero de 2021, de DZone: https://dzone.com/articles/why-edge-computing-is-so-important-to-industry-40

Kafka python. (s.f.). *Kafka python.* Recuperado el 27 de Julio de 2020, de Kafka python: https://kafka-python.readthedocs.io/en/master/index.html

Leshan, E. (17 de Junio de 2020). *GitHub.* Recuperado el 3 de Julio de 2020, de GitHub: https://github.com/eclipse/leshan/wiki/F.A.Q.

Microsoft. (s.f.). *Microsoft.* Recuperado el 1 de Febrero de 2021, de Microsoft: https://www.microsoft.com/en-us/research/project/live-video-analytics/

MongoDB. (s.f.). *MongoDB*. Recuperado el 14 de Septiembre de 2020, de MongoDB: https://docs.mongodb.com/manual/reference/method/db.collection.countDocuments/

No Magic Documentation. (27 de Diciembre de 2016). *No Magic Documentation*. Recuperado el 25 de Agosto de 2020, de No Magic Documentation: https://docs.nomagic.com/display/MD184/MagicDraw+Documentation

OMA SpecWorks. (s.f.). *OMA SpecWorks*. Recuperado el 13 de Enero de 2021, de OMA SpecWorks: https://omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/

Open Mobile Alliance. (10 de Julio de 2018). *Open Mobile Alliance*. Recuperado el 30 de Julio de 2020, de Open Mobile Alliance: http://www.openmobilealliance.org/release/LightweightM2M/V1_1-20180710-A/OMA-TS-LightweightM2M_Core-V1_1-20180710-A.pdf

Open Mobile Alliance. (s.f.). *Open Mobile Alliance*. Recuperado el 1 de Agosto de 2020, de Open Mobile Alliance: http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html

OpenCV. (10 de Octubre de 2018). *OpenCV.* Recuperado el 16 de Julio de 2020, de OpenCV: https://answers.opencv.org/question/200869/e1101module-cv2-has-no-imread-member/

OpenCV. (16 de Julio de 2020). *OpenCV.* Recuperado el 16 de Julio de 2020, de OpenCV: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

OpenCV. (s.f.). *OpenCV*. Recuperado el 14 de Enero de 2021, de OpenCV: https://opencv.org/

Oracle . (s.f.). *Oracle* . Recuperado el 10 de Noviembre de 2020, de Oracle : https://docs.oracle.com/javase/8/docs/api/java/nio/file/Files.html#readAllBytes-java.nio.file.Path-

Oracle. (s.f.). *Oracle*. Recuperado el 10 de Noviembre de 2020, de Oracle: https://docs.oracle.com/javase/9/docs/api/java/io/InputStream.html#readAllBytes--

Pabón, P. E. (31 de Julio de 2018). *SMARTSOFT*. Recuperado el 14 de Enero de 2021, de SMARTSOFT: https://smartsoftcolombia.com/portal/index.php/blog/49-rup

Platzi. (2019). Recuperado el 14 de Enero de 2021, de Platzi: https://platzi.com/blog/7-razones-mongodb/

Pysource. (27 de Junio de 2019). *Pysource.* Recuperado el 20 de Julio de 2020, de Pysource: https://pysource.com/2019/06/27/yolo-object-detection-using-opencv-with-python/

Python Docs. (s.f.). *Python Docs*. Recuperado el 4 de Diciembre de 2020, de Python Docs: https://docs.python.org/3/tutorial/errors.html

Python Software Foundation. (16 de Julio de 2020). *Docs Python.* Recuperado el 16 de Julio de 2020, de Docs Python: https://docs.python.org/3/library/copy.html

Radio Televisión Española. (28 de Noviembre de 2020). *rtve.* Recuperado el 21 de Enero de 2021, de rtve: https://www.rtve.es/alacarta/videos/zoom-net/5g-maritime-moviles-plegables-cortometrajes-moviles/5726693/

Rao, A. S. (s.f.). *Timber.* Recuperado el 26 de Julio de 2020, de Timber: https://timber.io/blog/hello-world-in-kafka-using-python/

Rathor, A. (25 de Mayo de 2020). *Towards Data Science*. Recuperado el 8 de Noviembre de 2020, de Towards Data Science: https://towardsdatascience.com/run-multiple-services-in-single-docker-container-using-supervisor-b2ed53e3d1c0

Redhat. (s.f.). *Redhat*. Recuperado el 13 de Enero de 2021, de https://www.redhat.com/es/topics/containers/what-is-docker

Santos, P. R. (22 de Septiembre de 2020). *Think Big/Empresas.* Recuperado el 25 de Enero de 2021, de Think Big/Empresas: https://empresas.blogthinkbig.com/breve-historia-de-internet-de-las-cosas-iot/

Shanmugam, K. (26 de Septiembre de 2018). *Medium*. Recuperado el 7 de Noviembre de 2020, de Medium: https://medium.com/@karthi.net/how-to-run-multiple-services-in-a-docker-container-5919fcc981a6

Shokeen, M. (9 de Marzo de 2017). *Envatotuts+*. Recuperado el 11 de Septiembre de 2020, de Envatotuts+: https://code.tutsplus.com/es/tutorials/using-the-requests-module-in-python--cms-28204

StackOverflow. (25 de Enero de 2017). *StackOverflow.* Recuperado el 20 de Julio de 2020, de StackOverflow:

https://stackoverflow.com/questions/10624937/convert-datetime-object-to-a-string-of-date-only-in-python
StackOverflow. (18 de Agosto de 2017). *StackOverflow.* Recuperado el 26 de Julio de 2020, de StackOverflow: https://stackoverflow.com/questions/17967320/python-opencv-convert-image-to-byte-string
StackOverflow. (12 de Octubre de 2017). *StackOverflow*. Recuperado el 17 de Septiembre de 2020, de StackOverflow: https://es.stackoverflow.com/questions/109040/c%C3%B3mo-puedo-insertar-delay-en-python
StackOverflow. (3 de Mayo de 2020). *StackOverflow.* Recuperado el 16 de Julio de 2020, de StackOverflow: https://stackoverflow.com/questions/678236/how-to-get-the-filename-without-the-extension-from-a-path-in-python
StackOverflow. (5 de Enero de 2020). *StackOverflow.* Recuperado el 27 de Julio de 2020, de StackOverflow: https://stackoverflow.com/questions/17170752/python-opencv-load-image-from-byte-string
StackOverflow. (19 de Febrero de 2020). *StackOverflow.* Recuperado el 30 de Julio de 2020, de StackOverflow: https://stackoverflow.com/questions/42525139/maven-build-compilation-error-failed-to-execute-goal-org-apache-maven-plugins
StackOverflow. (31 de Julio de 2020). *StackOverflow*. Recuperado el 5 de Agosto de 2020, de StackOverflow: https://stackoverflow.com/questions/2418485/how-do-i-convert-a-byte-array-to-base64-in-java
StackOverflow. (27 de Abril de 2020). *StackOverflow*. Recuperado el 14 de Octubre de 2020, de StackOverflow: https://stackoverflow.com/questions/61466799/docker-failed-to-establish-a-new-connection-errno-111-connection-refused
StackOverflow. (9 de Septiembre de 2020). *StackOverflow*. Recuperado el 8 de Diciembre de 2020, de StackOverflow: https://stackoverflow.com/questions/40928205/python-opencv-image-to-byte-string-for-json-transfer
StackOverFlow. (5 de Mayo de 2020). *StackOverFlow*. Recuperado el 11 de Septiembre de 2020, de StackOverFlow: https://es.stackoverflow.com/questions/352451/typeerror-nonetype-object-is-not-subscriptable
StackOverFlow. (20 de Julio de 2020). *StackOverFlow*. Recuperado el 20 de Noviembre de 2020, de StackOverFlow: https://stackoverflow.com/questions/50807514/kafka-python-consumers-running-in-parallel-threads
StackOverflow. (s.f.). *StackOverflow.* Recuperado el 16 de Julio de 2020, de StackOverflow: https://stackoverflow.com/questions/44663347/python-opencv-reading-the-image-file-name
StackOverflow. (s.f.). *StackOverflow*. Recuperado el 5 de Agosto de 2020, de StackOverflow: https://stackoverflow.com/questions/858980/file-to-byte-in-java

StackOverFlow. (s.f.). *StackOverFlow*. Recuperado el 11 de Septiembre de 2020, de StackOverFlow: https://stackoverflow.com/questions/47668507/how-to-store-images-in-mongodb-through-pymongo

Torres, G. (15 de Febrero de 2019). *returngis.* Recuperado el 30 de Junio de 2020, de returngis: https://www.returngis.net/2019/02/ejecutar-aplicaciones-multi-contenedor-con-docker-compose/

Torres, G. (21 de Febrero de 2019). *returngis.* Recuperado el 30 de Junio de 2020, de returngis: https://www.returngis.net/2019/02/aplicaciones-multi-contenedor-en-un-cluster-con-swarm/

Unipython. (s.f.). *Unipython*. Recuperado el 16 de Septiembre de 2020, de Unipython: https://unipython.com/como-ejecutar-una-funcion-desde-un-enlace-link-html-con-java-script/

Universidad Politécnica de Valencia. (s.f.). Recuperado el 13 de Enero de 2021, de http://personales.upv.es/rmartin/cursoJava/Java/Introduccion/PrincipalesCaracteristicas.htm

Universitat de Girona. (s.f.). Recuperado el 14 de Enero de 2021, de Universitat de Girona: http://ima.udg.edu/~sellares/EINF-ES2/Present1011/MetodoPesadesDocumentacio.pdf

Vallecillo, A. (1 de Octubre de 2018). *Modelado y Diseño de Software.* Recuperado el 13 de Enero de 2021, de Modelado y Diseño de Software.

Visual Studio Code. (10 de Octubre de 2018). *code.visualstudio.* Recuperado el 16 de Julio de 2020, de code.visualstudio: https://code.visualstudio.com/docs/python/linting

Visus, A. (Octubre de 2020). *ESIC*. Recuperado el 14 de Enero de 2021, de ESIC: https://www.esic.edu/rethink/tecnologia/para-que-sirve-python

W3School. (s.f.). *w3School.* Recuperado el 16 de Julio de 2020, de w3School: https://www.w3schools.com/python/python_classes.asp

W3School. (s.f.). *W3School*. Recuperado el 9 de Septiembre de 2020, de W3School: https://www.w3schools.com/python/python_mongodb_getstarted.asp

W3School. (s.f.). *W3School*. Recuperado el 14 de Septiembre de 2020, de W3School: https://www.w3schools.com/html/default.asp

W3School. (s.f.). *W3School*. Recuperado el 16 de Septiembre de 2020, de W3School: https://www.w3schools.com/howto/howto_js_redirect_webpage.asp

W3School. (s.f.). *W3School*. Recuperado el 17 de Septiembre de 2020, de W3School: https://www.w3schools.com/js/js_ajax_http_send.asp

Wikipedia. (12 de Noviembre de 2020). *Fundación Wikipedia*. Recuperado el 14 de Enero de 2021, de Fundación Wikipedia: https://es.wikipedia.org/wiki/Firebase

Wikipedia. (7 de Noviembre de 2020). *Wikipedia*. Recuperado el 13 de Enero de 2021, de https://en.wikipedia.org/wiki/Constrained_Application_Protocol

Wikipedia. (10 de Diciembre de 2020). *Wikipedia Foundation*. Recuperado el 13 de Enero de 2021, de Wikipedia Foundation: https://en.wikipedia.org/wiki/OMA_LWM2M

Wikipedia. (8 de Diciembre de 2020). *Wikipedia Foundation*. Recuperado el 14 de Enero de 2021, de Wikipedia Foundation: https://es.wikipedia.org/wiki/OpenCV

Wikipedia. (11 de Enero de 2021). *Wikipedia Foundation*. Recuperado el 13 de Enero de 2021, de Wikipedia Foundation: https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

Wikipedia Foudation. (5 de Noviembre de 2020). *Wikipedia*. Recuperado el 14 de Enero de 2021, de Wikipedia: https://es.wikipedia.org/wiki/Programaci%C3%B3n_extrema

Wikipedia Foundation. (7 de Agosto de 2020). *Wikipedia*. Recuperado el 14 de Enero de 2021, de Wikipedia: https://es.wikipedia.org/wiki/Herramienta_CASE

Wikipedia Foundation. (11 de Octubre de 2020). *Wikipedia*. Recuperado el 14 de Enero de 2021, de Wikipedia: https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado

Wikipedia Foundation. (22 de Agosto de 2020). *Wikipedia*. Recuperado el 14 de Enero de 2021, de Wikipedia: https://en.wikipedia.org/wiki/Interaction_Flow_Modeling_Language

Wikipedia Foundation. (17 de Septiembre de 2020). *Wikipedia*. Recuperado el 14 de Enero de 2021, de Wikipedia: https://es.wikipedia.org/wiki/Desarrollo_en_cascada

Wikipedia Foundation. (26 de Noviembre de 2020). *Wikipedia* . Recuperado el 21 de Enero de 2021, de Wikipedia: https://es.wikipedia.org/wiki/Algoritmo_You_Only_Look_Once_(YOLO)

Wikipedia Foundation. (1 de Diciembre de 2020). *Wikipedia Foundation.* Recuperado el 14 de Enero de 2021, de Wikipedia Foundation: https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software#Metodolog%C3%ADas_de_desarrollo_de_software

Wikipedia Foundation. (7 de Junio de 2020). *Wikipedia Foundation.* Recuperado el 14 de Enero de 2021, de Wikipedia Foundation: https://es.wikipedia.org/wiki/Especificaci%C3%B3n_de_requisitos_de_software

Wikipedia Foundation. (22 de Enero de 2021). *Wikipedia.* Recuperado el 1 de Febrero de 2021, de Wikipedia: https://en.wikipedia.org/wiki/Type-length-value

Wikipedia Foundation. (1 de Febrero de 2021). *Wikipedia .* Recuperado el 1 de Febrero de 2021, de Wikipedia: https://en.wikipedia.org/wiki/OpenVINO

Wikipedia Foundation. (8 de Enero de 2021). *Wikipedia Foundation.* Recuperado el 14 de Enero de 2021, de Wikipedia Foundation: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)#Caracter%C3%ADsticas_de_Scrum

YOLO. (s.f.). Recuperado el 14 de Enero de 2021, de YOLO: https://pjreddie.com/darknet/yolo/

Apéndice A

Manual de Instalación

En el manual de instalación se indicará paso a paso los procedimientos necesarios para realizar una instalación correcta del sistema. Este manual solo es apto para el entorno en el que se ha desarollado, Linux.

Requerimientos:

- Docker CE - https://docs.docker.com/engine/install/debian/

Jerarquía del Proyecto:

- *IoT_Database*
- *IoT_Monitor*
- *IoT_Recovery*
- *IoT_Register*
- *IoT_ShadowApplications*
- *IoT_Web*
- *IoT_LeshanServer*
- *IoT_LeshanClient*
- *IoT_ObjectDetectionApp*
- *IoT_ImagesAPIDB*
- *IoT_ImagesWeb*
- *docker-compose-all-2.yml*

Disponiendo de la jerarquía que se puede apreciar en la lista anterior, primero debemos crear las imágenes docker correspondientes a cada módulo. Para ello se accede a la carpeta raíz de cada módulo y ejecutamos los siguientes comandos en cada caso:

- docker build --tag= iotdatabase .
- docker build --tag= iotregister .
- docker build --tag= iotweb .
- docker build --tag= iotshadowapp .
- docker build --tag= iotrecovery .
- docker build --tag= iotimagesapidb .
- docker build --tag= iotimagesweb .
- docker build --tag= iotobjectdetection .
- docker build --tag= leshanmonitor .
- docker build --tag= leshanclient .
- docker build --tag= leshanserver .

Nótese que los nombres del tag están en concordancia con el fichero docker-compose-all-2.yml, si se desean otros nombres, también deben modificarse en el fichero.

Una vez obtenidas todas las imágenes, el siguiente paso es desplegarlas sobre Docker Swarm, para ello debemos abrir una terminal y ejecutamos el siguiente comando:

- docker swarm init

Con este comando inicializamos el enjambre docker.

Una vez inicializado, el siguiente paso es desplegar las imágenes creadas con anterioridad. Para ello, en la terminal nos dirigimos a la ruta donde se encuentra el fichero docker-compose-all-2.yml y procedemos a ejecutar el siguiente comando:

- docker stack deploy -c docker-compose-all-2.yml reliableiot

Siendo reliableiot, el nombre de la red por la que se comunicarán nuestros módulos.

Una vez realizado estos pasos debemos tener algo parecido a lo que se aprecia en la Figura M.I.1:

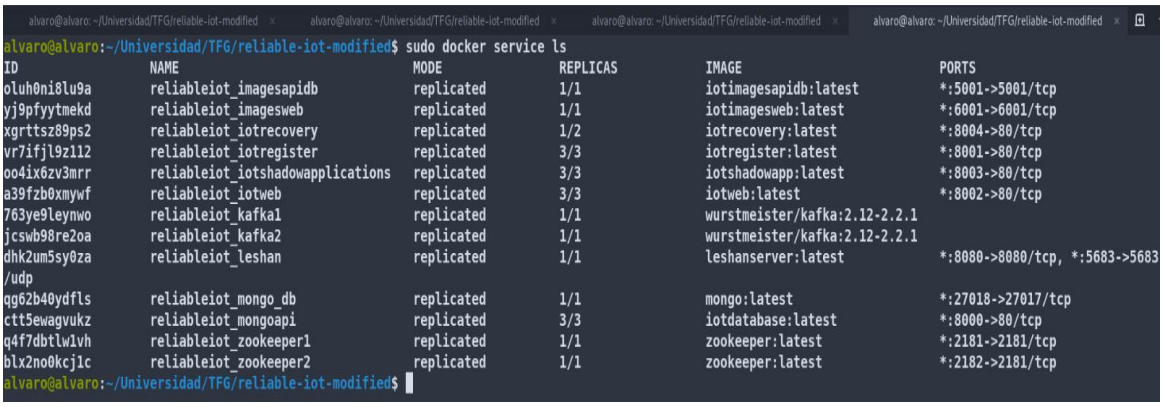

Figura M.I. 1: Servicios Desplegados

Una vez la instalación completada, podemos utilizar el sistema tal y como se especifica en el manual de uso.

Apéndice B

Manual de Uso

El manual de usuario contendrá toda la información para el uso de la aplicación por cualquier tipo de usuario independientemente de su familiaridad con aplicaciones de este tipo. Será un manual con estilo tutorial, en el que se explicarán los pasos a seguir para realizar cada una de las funciones del sistema.

Una vez el sistema esté instalado, debemos hacer lo siguiente:

Abrimos el navegador y nos dirigimos a la dirección 127.0.0.1:8002 (esa dirección corresponde con el módulo *IoT_Web)*.

Una vez ahí, nos registramos en el sistema.

Tras el registro nos mostrará una página principal en la que nos crearemos nuestro *shadow device*. El resultado debe ser similar a la Figura M.U. 1

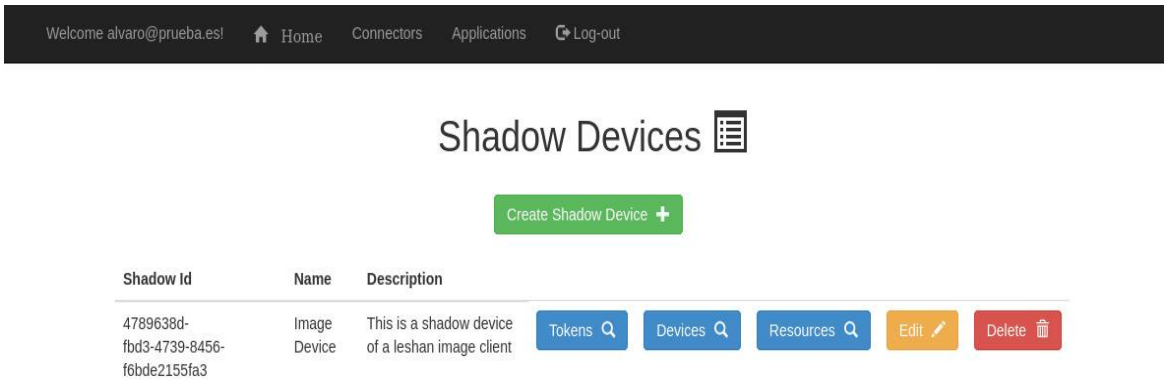

Figura M.U. 1: Dispositivo Shadow Creado

Cuando nos hayamos creado nuestro *shadow device,* tendremos que crear el monitor en la pestaña *Connectors* y un token que usaremos para el registro de nuestro dispositivo físico.

Tras esto ya podemos registrar nuestro cliente *leshan*.

Al pulsar el botón ***Device*** de un dispositivo *shadow*, se mostrará una página (ver Figura M.U. 2) con los dispositivos físicos registrados en el sistema pertenecientes al dispositivo Shadow respectivo.

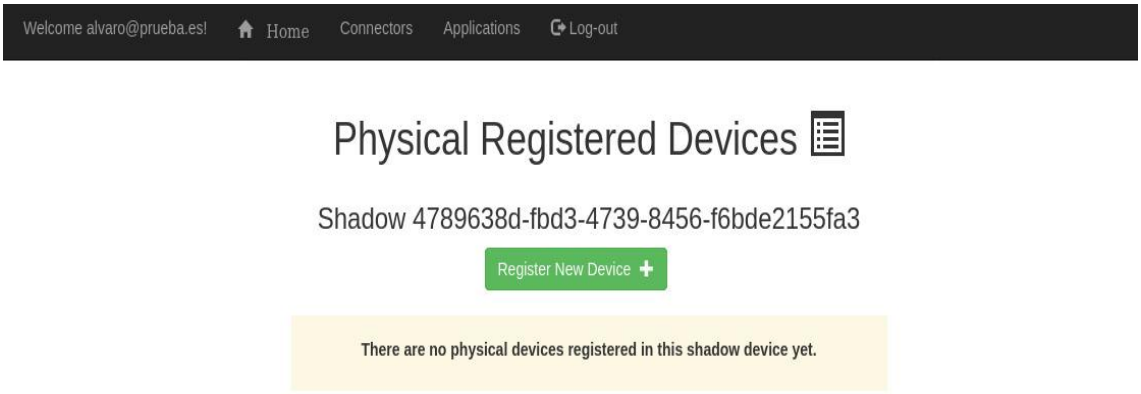

Figura M.U. 2 : Dispositivos Físicos Registrados (Vacío)

Para registrar nuestro cliente pulsamos el botón ***Register New Device*** que nos mostrará un formulario a cumplimentar. Tras completar el formulario (ver Figura M.U. 3), pulsamos el botón ***Create*** y nuestro dispositivo ya está dado de alta en el sistema (ver Figura M.U. 4).

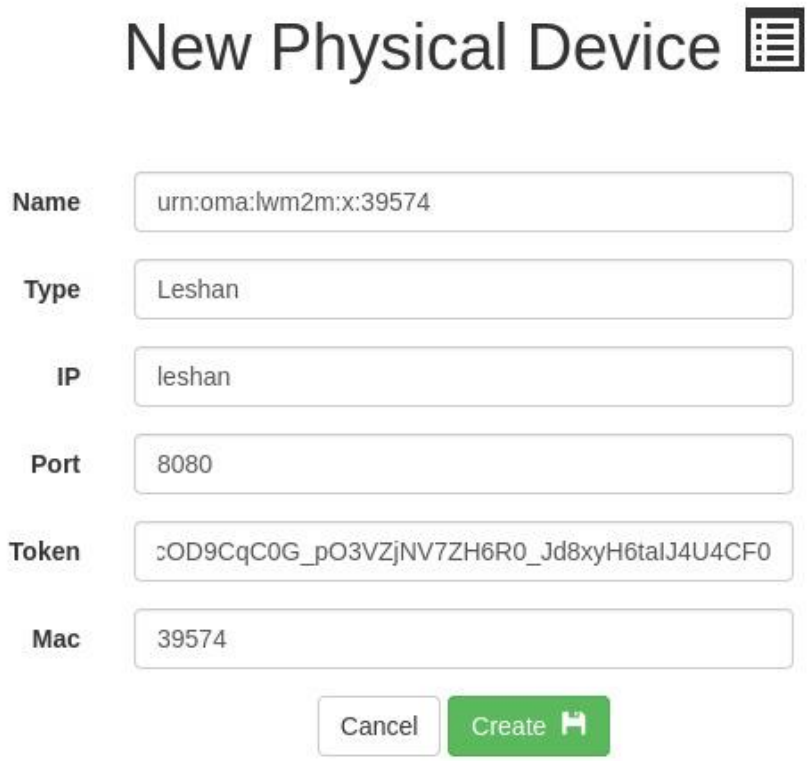

Figura M.U. 3 Formulario de Registro Dispositivo

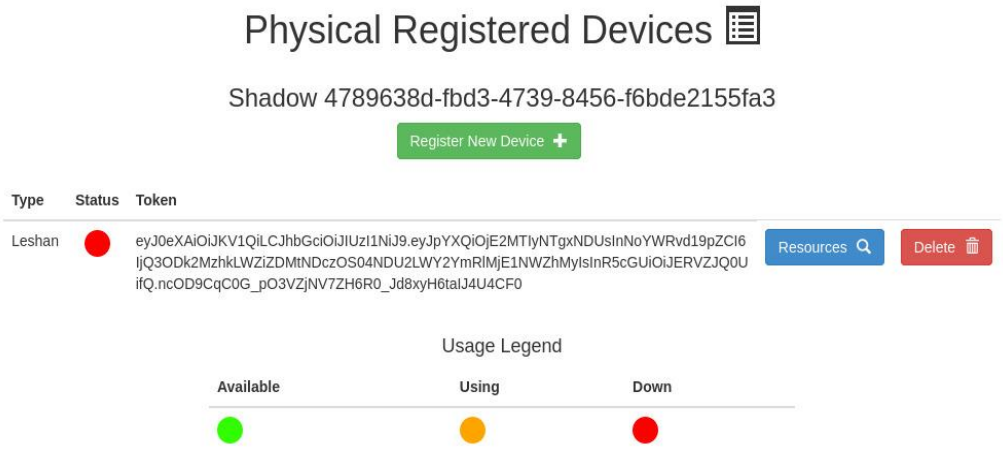

Figura M.U. 4: Dispositivo Registrado (Down)

Tras el registro, nos dirigimos de nuevo a la terminal para desplegar nuestro cliente leshan. Para ello introducimos el siguiente comando:

- docker service create --name reliableiot_leshanclient --network reliableiot leshanclient

Como se puede apreciar en la Figura M.U.5, nuestro cliente ya está desplegado en la arquitectura. Si refrescamos la web, podremos observar que nuestro cliente ya está disponible (color Verde), tal y como se muestra en la Figura M.U.6.

Figura M.U. 5: Despliegue de Cliente Leshan

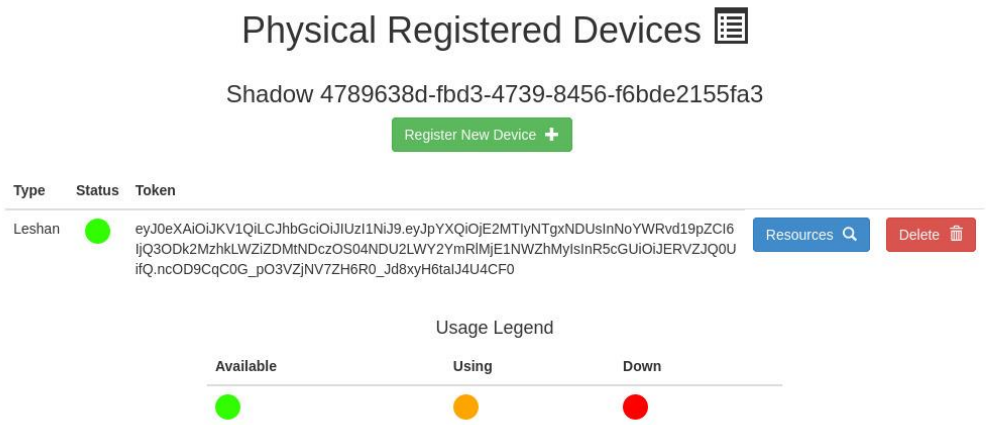

Figura M.U. 6: Dispositivo Registrado (Available)

Después de registrar nuestro dispositivo, procederemos a desplegar nuestra aplicación de detección de objetos. Para desplegarla, volvemos a la consola y escribimos el siguiente comando:

- docker service create --name reliableiot_objectdetection --network reliableiot iotobjectdetection

Figura M.U. 7: Aplicación Detección Objetos Desplegada

Como se puede apreciar en la Figura M.U. 7, la aplicación ya está desplegada y solo queda esperar a que se envíen las imágenes.

Para poder ver el resultado del análisis de imágenes abrimos una pestaña en nuestro navegador e ingresamos la siguiente url: [127.0.0.1:6001](127.0.0.1:6001)

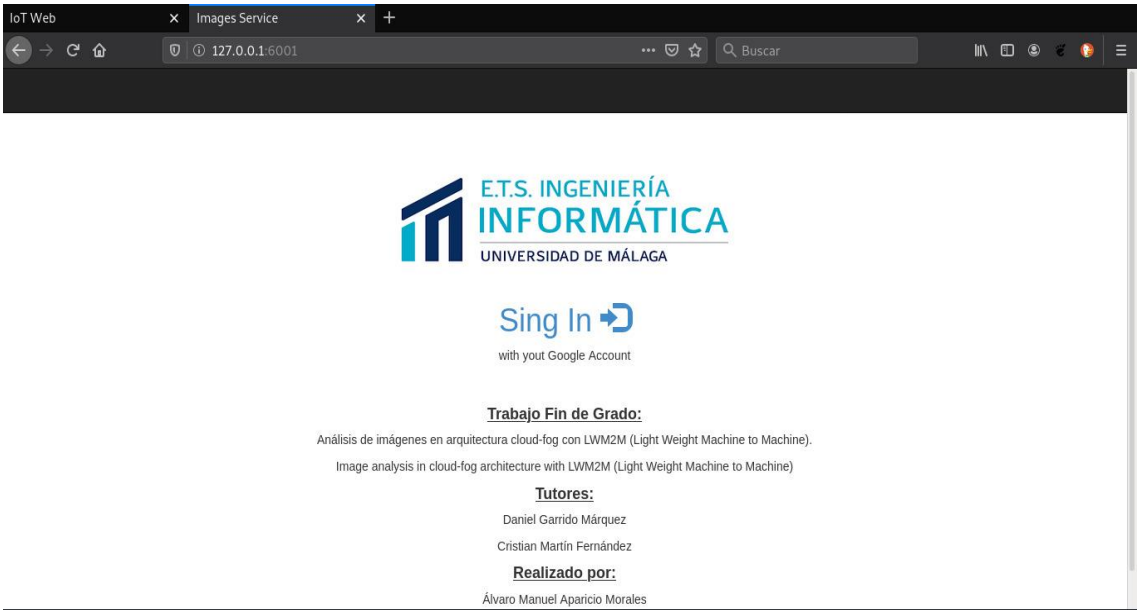

Figura M.U. 8: Inicio Aplicación Web

Al pulsar sobre ***Sing In***, se nos abrirá una ventana para iniciar sesión con nuestra cuenta de Google. Una vez hecho esto nos mostrará una página principal donde se podrán observar las imágenes procesadas (Ver Figura M.U.9, Figura M.U. 10 y Figura M.U.11).

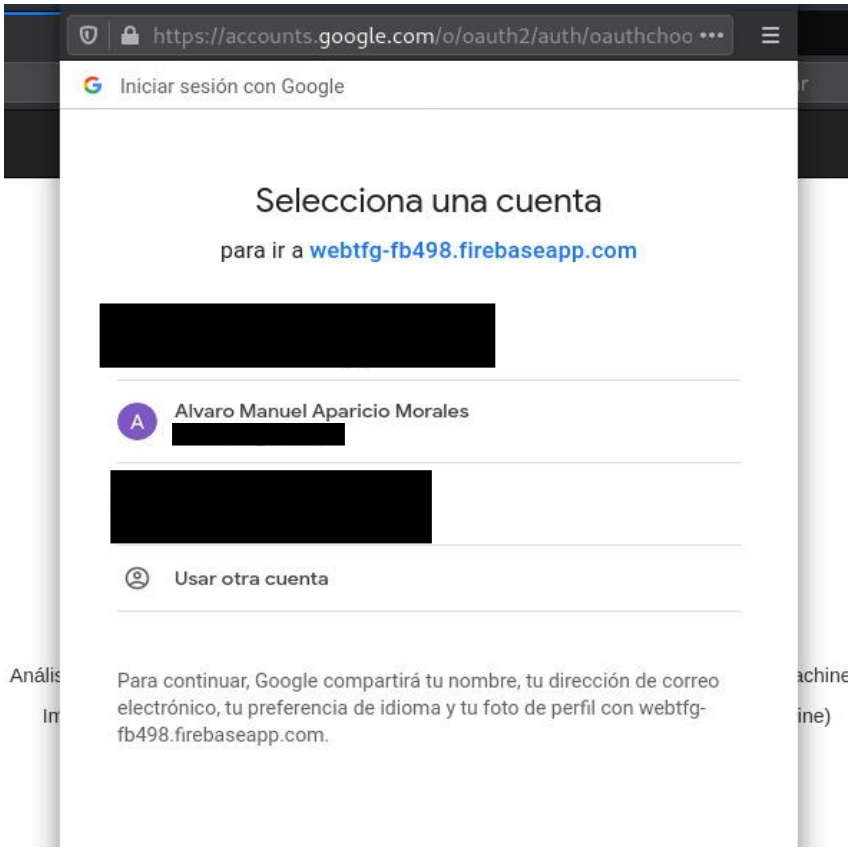

Figura M.U. 9: Inicio Sesión Google

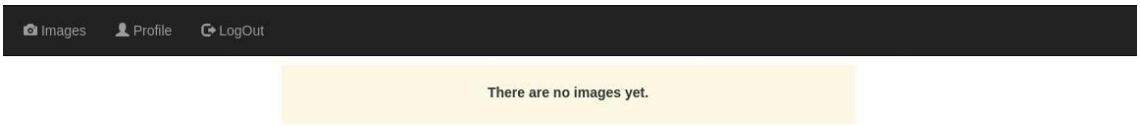

Figura M.U. 10: Vista Principal Aplicación (Sin Imágenes)

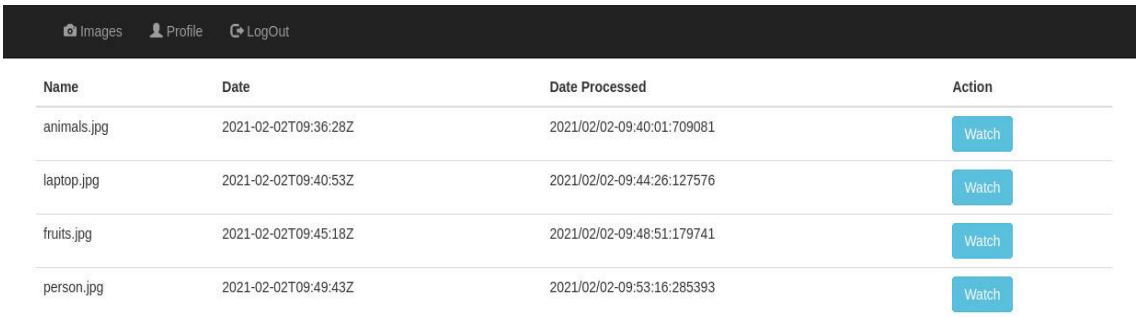

Figura M.U. 11: Vista Principal Aplicación (Con Imágenes)

Si deseamos ver la imagen, pulsamos en el botón ***Watch*** de color azul. (Ver Figura M.U.12).

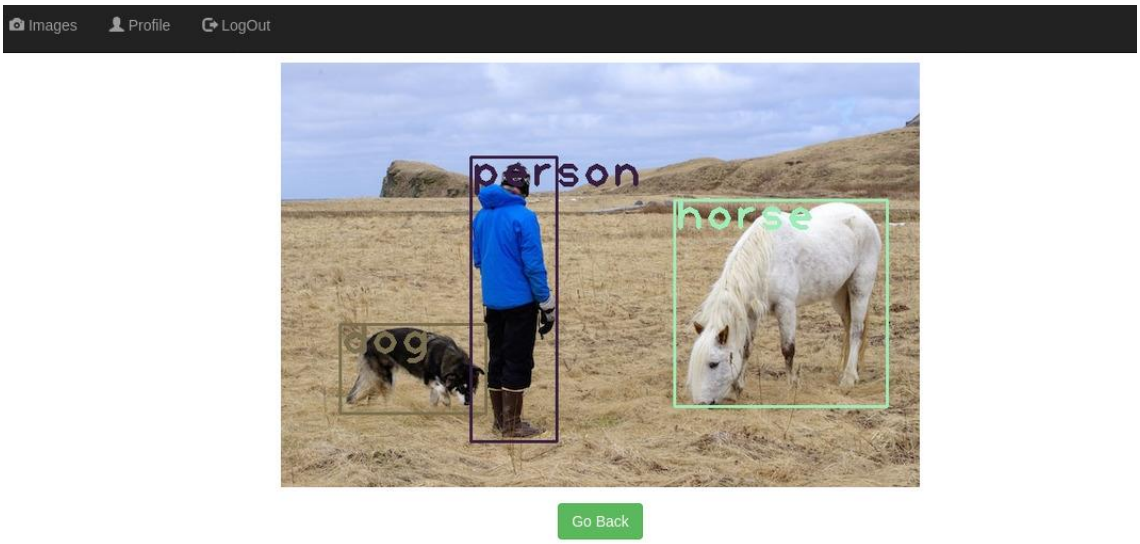

Figura M.U. 12: Resultado Análisis

En la barra de navegación podremos, cerrar sesión o acceder a nuestro perfil, que nos permitirá eliminar nuestro perfil de la aplicación (Ver Figura M.U.13).

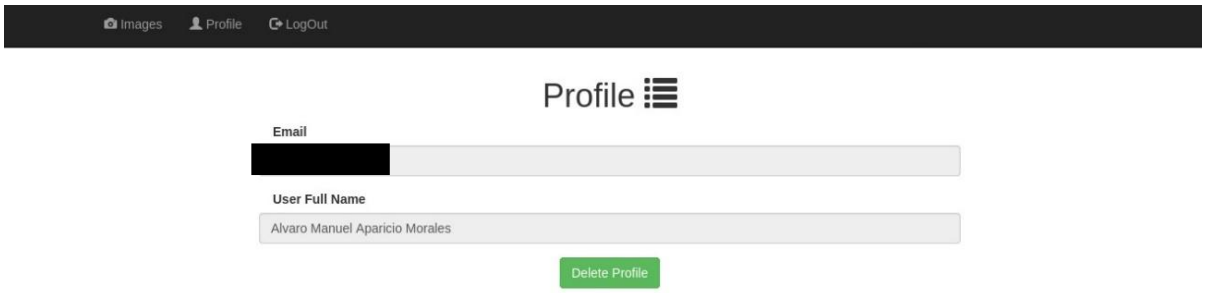

Figura M.U. 13: Vista Perfil Personal

Apéndice C

Notas

[i] *CoAP (Constraint Application Protocol): es un protocolo especializado implementado en la capa de aplicación para ser usado en dispositivos de Internet con recursos limitados.* (Wikipedia, 2020)
[ii] *PyMongo: es una librería de Python para poder conectarnos a una base de datos MongoDB.* (Cuervo, 2016)

[iii] *CASE (Cumputer Aided Software Engineering, Ingeniería de Software Asistida por Computadora): son diversas aplicaciones informáticas o programas informáticos destinadas a aumentar el balance en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software.* (Wikipedia Foundation, 2020)
[iv] *UML (Unified Modeling Language, Lenguaje Unificado de Modelado): Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad* (Wikipedia Foundation, 2020)
[v] *IFML (Interaction Flow Modeling Language, Lenguaje de modelado de flujo de interacción): es un lenguaje de modelado estandarizado en el campo de la ingeniería de software.*
[vi] *Docker Swarm* : *Es una herramienta integrada en el ecosistema de Docker que permite la gestión de un clúster de servidores. Pone a nuestra disposición una API con la que podemos administrar las tareas y asignación de recursos de cada contenedor dentro de cada una de las máquinas. Dicha API nos permite gestionar el clúster como si se tratase de una sola máquina Docker.* (Gómez, s.f.)
[vii] *Docker Compose: Es una herramienta de Docker para la definición y ejecución de aplicaciones multicontenedor. Mediante el archivo YAML se configuran los servicios de la aplicación y con un simple comando se produce el despliegue.* (Docker, s.f.)
[viii] *Desarrollo en Cascada: Es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior.* (Wikipedia Foundation, 2020)
[ix] *RUP (Rational Unified Process, Proceso de Desarrollo Unificado): Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.* (Pabón, 2018)
[x] *XP (eXtrem Programming, Programación Extrema): es una metodología de desarrollo de la ingeniería de software que se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.* (Wikipedia Foudation, 2020)
[xi] *Enabler Test Specfication for LightweightM2M: Es un documento en formato pdf cuyo propósito es el de proporcionar los casos de pruebas para LightweightM2M. Puede consultarse en la siguiente url: http://www.openmobilealliance.org/release/LightweightM2M/ETS/OMA-ETS-LightweightM2M-V1_0-20140226-C.pdf*
[xii] *Gherkin: Es un lenguaje utilizado en la descripción de pruebas en Cucumber. Martin Fowler lo clasifica como Business Readable DSL (Lenguaje Específico de Dominio Legible por Negocio.*

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga