UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de Ingeniera de Sistemas e Ingeniero de Sistemas

PROYECTO TÉCNICO:

"DISEÑO Y DESARROLLO DE UN SISTEMA DE PROTOTIPO DE RECONOCIMIENTO FACIAL SOBRE INFRAESTRUCTURA FOG COMPUTING EN UNA ARQUITECTURA DE MICROSERVICIOS MONTADA EN CONTENEDORES DOCKER EJECUTADAS EN UNA INSTANCIA DE INFRAESTRUCTURA DE NUBE DISTRIBUIDA EN OPENNEBULA, APLICACIÓN MÓVIL ANDROID A LOS MEDIOS DE TRANSPORTE PÚBLICO"

AUTORES:

ANITA MELANIA SICHIQUE PILLACELA

MAICOLY EMANUEL GUERRERO ZHUNIO

TUTOR:

PABLO LEONIDAS GALLEGOS SEGOVIA, PhD.

CUENCA - ECUADOR

2020

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Anita Melania Sichique Pillacela con documento de identificación N° 1900511641 y Maicoly Emanuel Guerrero Zhunio con documento de identificación Nº 1400634224, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación: **"DISEÑO Y DESARROLLO DE UN SISTEMA PROTOTIPO DE RECONOCIMIENTO FACIAL SOBRE INFRAESTRUCTURA FOG COMPUTING EN UNA ARQUITECTURA DE MICROSERVICIOS MONTADA EN CONTENEDORES DOCKER EJECUTADAS EN UNA INSTANCIA DE INFRAESTRUCTURA DE NUBE DISTRIBUIDA EN OPENNEBULA, APLICACIÓN MOVIL ANDROID A LOS MEDIOS DE TRANSPORTE PÚBLICO"**, mismo que ha sido desarrollado para optar por el título de: *Ingeniera de Sistemas e Ingeniero de Sistemas*, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores, nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, marzo del 2020

Anita Melania Sichique Pillacela
C.I.1900511641

Maicoly Emanuel Guerrero Zhunio
C.I. 1400634224

CERTIFICACIÓN

Yo, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **"DISEÑO Y DESARROLLO DE UN SISTEMA PROTOTIPO DE RECONOCIMIENTO FACIAL SOBRE INFRAESTRUCTURA FOG COMPUTIG EN UNA ARQUITECTURA DE MICROSERVICIOS MONTADAS EN CONTENEDORES DOCKER EJECUTADAS EN UNA INSTANCIA DE INFRAESTRUCTURA DE NUBE DISTRIBUIDA EN OPENNEBULA, APLICACIÓN MOVIL ANDROID A LOS MEDIOS DE TRANSPORTE PÚBLICO"**, realizado por Anita Melania Sichique Pillacela y Maicoly Emanuel Guerrero Zhunio, obteniendo el *Proyecto Técnico*, que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana.

Cuenca, marzo del 2020

Ing. Pablo Leonidas Gallegos Segovia, PhD.

C.I 0102593589

DECLARATORIA DE RESPONSABILIDAD

Nosotros, Anita Melania Sichique Pillacela con documento de identificación N° 1900511641 y Maicoly Emanuel Guerrero Zhunio con documento de identificación Nº 1400634224, autores del trabajo de titulación: **"DISEÑO Y DESARROLLO DE UN SISTEMA PROTOTIPO DE RECONOCIMIENTO FACIAL SOBRE INFRAESTRUCTURA FOG COMPUTIG EN UNA ARQUITECTURA DE MICROSERVICIOS MONTADAS EN CONTENEDORES DOCKER EJECUTADAS EN UNA INSTANCIA DE INFRAESTRUCTURA DE NUBE DISTRIBUIDA EN OPENNEBULA, APLICACIÓN MOVIL ANDROID A LOS MEDIOS DE TRANSPORTE PÚBLICO"**, certificamos que el total contenido del *Proyecto Técnico,* es de nuestra exclusiva responsabilidad y autoría.

Cuenca, marzo del 2020

Anita Melania Sichique Pillacela
C.I.1900511641

Maicoly Emanuel Guerrero Zhunio
C.I. 1400634224

AGRADECIMIENTOS

Queremos hacer un profundo agradecimiento al grupo de investigación de Cloud Computing, Smart Cities & High Performance Computing por pertenecer a este grupo, con el cual nos brindaron un gran apoyo.

A nuestros compañeros y docentes, que pasaron a ser una parte de apoyo dentro de nuestro carrara de aprendizaje.
Nuestro más sincero agradecimiento a nuestro tutor al PhD Pablo Gallegos quien fue nuestro guía incondicionalmente durante proceso de investigación

Maicoly Emanuel Guerrero – Anita Melania Sichique

DEDICATORIA

A mi bendito padre Dios y mi virgencita del Cisne que por ellos han logrado hacer lo que soy ahora, gracias por siempre guiarme por los buenos caminos y alejarme de la maldad y sobre todo brindarme la salud, y buenas decisiones para la culminación de mi tesis.

Dedico este trabajo a mi madre Nancy Zhunio por el constante apoyo ya que sin ella no sería posible la culminación de este proyecto.

A mi hijito Ethan Morocho en el cielo; te amo muchísimo; por él fue posible la codificación e implementación del prototipo y todo el sistema en microservicios; sin la presencia de él quien sabe dónde hubiera estado, aunque Dios decidió tenerte con él siempre, todos los días estas en mis pensamientos, esto es gracias ti pequeño Maicky.

A mi tutor de tesis el Dr. Pablo Gallegos Segovia por brindarnos el apoyo incondicional y guiarnos en todo el proceso y seguimiento del proyecto.

A mi compañera Anita Sichique por los consejos y ser la imagen del nunca rendirse, fue un gran apoyo al construir y culminar este proyecto.

A mis hermanos Anthony, Geancarlos, Yadira por la paciencia y ayuda en todas las noches largas de desarrollo y consejos.

A mi promesa pasada, para Fabiola Jara, aunque nuestras vidas no se unieron uno de mis objetivos fue este, me ayudaste con mis tareas y muchas cosas durante mi vida de estudiante Gracias!.

Maicoly Emanuel Guerrero Zhunio

DEDICATORIA

Mi más sincero agradecimiento a Dios por regalar la familia que me ha tocado, estoy tan orgullosa en pertenecer a ella oh, mi Dios padre eterno de amor que me has enseñado a confiar en lo imposible, que con tu infinito amor todo se puede, por el amor que nos has dado y aprendió a perdonar.

A mis padres por regalarme la vida y cuidarme en los primeros pasos de mi vida y ensenarme sus valores con gran humildad y bondad y gracias a ellos estoy culminado una meta más en mi vida ya que fueron un apoyo durante esta eta.

A mi hermana Laura, aparte de ser mi herma fue una gran amiga que siempre estado en los momentos más difíciles de la vida, que se me presento durante este proceso de mi carrera ya que ella fue un pilar fundamente de apoyo.

Mi agradecimiento al Ing. Remigio Hurtado gracias a sus enseñanzas asido un ex lente como persona y docente en el proceso de su enseñanza.

A todos mis compañeros y docentes los que brindaron un apoyo incondicional durante este proceso, en especial al Ing. Pablo Gallegos que nos acogió sin dificultad a ser nuestro tutor. A mi compañero del proyecto Maicoly Guerrero por su apoyo

Anita Melania Sichique Pillacela

ÍNDICE GENERAL

Índice de Imágenes

Índice de Tablas

1. RESUMEN

La siguiente investigación se centró en el desarrollo de una aplicación móvil y una aplicación web para el reconocimiento facial de los criminales más buscados, de secuestradores y de personas desaparecidas, en la ciudad de Cuenca-Ecuador.

La innovación en esta aplicación con Fog Computing implicó el uso de microservicios con el fin de optimizar el procesamiento de inteligencia artificial; dichos sistemas independientes se encuentran dentro del servidor OpenNebula, el cual permitió ventajas como: reducción de costos, alta disponibilidad, además de caracterizarse por ser seguro y accesible para las personas que deseen trabajar en Cloud Computing.

La aplicación móvil desarrollada para el reconocimiento facial abarcó las siguientes funcionalidades: denuncia de casos, notificaciones, información de casos y wanted-face. Por otro lado, la aplicación web contó con los siguientes módulos: dashboard, administración y mantenimientos. Con la implementación de los microservicios en el desarrollo de este aplicativo se logró la optimización de los procesos de inteligencia artificial, debido a la escalabilidad y tolerancia a fallos de estos sistemas individuales es factible un mejor rendimiento en la velocidad de respuesta.

2. ABSTRACT

This research focused on the development of a mobile application and a web application on the facial recognition of the most wanted criminals, kidnappers and missing persons, in the city of Cuenca-Ecuador.

The innovation in this application with Fog Computing involved the use of microservices in order to optimize artificial intelligence processes; these independent systems were located within the OpenNebula server, which allowed advantages such as cost reduction, high availability, in addition to being characterized by being safe and accessible for people who wish to work in Cloud Computing.

The facial recognition mobile application covered the following functionalities: case reporting, the most wanted criminals, kidnappers and missing persons notifications, case information and wanted-face. On the other hand, the web application had the following modules: dashboard, administration and Crud. The implementation of microservices was

important in the development of this application. The optimization of artificial intelligence processes was achieved due to the scalability and fault tolerance of these individual systems, a better performance in response speed was feasible.

3. INTRODUCCION

El siguiente proyecto técnico tiene su fundamentación en la investigación aplicada cuya premisa consiste en la aplicación directa del conocimiento y la tecnología para la solución de un problema en específico.

Las preocupaciones, ante los problemas de inseguridad y violencia en Ecuador, se han convertido en tema de discusión para el área de las TIC's y el diseño de políticas, ya que afecta el "derecho a vivir" en las condiciones adecuadas que permitan el desarrollo de los ciudadanos. Esto preocupa no solamente a las instituciones policiales, sino a toda la colectividad en general.

La dificultad para encontrar a los criminales más buscados, la impunidad de los actos delictivos y los desafíos que enfrentan los investigadores frente a la desaparición de personas constituyen graves problemas que amenazan la seguridad de los ciudadanos ecuatorianos. Según el Ministerio del Interior, desde el 2014 hasta el 2017, en la provincia del Azuay se localiza un porcentaje de 10,30% de los criminales más buscados [40]., lo que representa una cifra importante que no se puede desestimar y que refleja una problemática social que pone en peligro a la población azuaya y, en particular, a los cuencanos que residen en la capital de la mencionada provincia. Asimismo, dentro de este espacio geográfico hay un 4,34% de personas desaparecidas [41]., lo cual pone en evidencia actividades delictivas asociadas al secuestro.

Considerando todo esto, la presente investigación académica centra su atención en el desarrollo de una aplicación móvil y una aplicación web con bases técnicas en los procesos de inteligencia artificial para facilitar el reconocimiento facial de los criminales más buscados, los secuestradores y las personas desparecidas, en Cuenca-Ecuador. Aunque ya existen programas de vigilancia asociados al Ecu 911, es importante destacar que la relevancia social de este proyecto técnico radica en el hecho de ser una solución factible a los problemas de inseguridad. Esta aplicación se

diseñó para ser implementada en los medios de transporte público de la ciudad de Cuenca.

Con el avance de la tecnología de hoy en día, se ha podido crear un sin número de aplicativos en donde el reconocimiento facial de las personas puede ser posible, usando dispositivos inteligentes como prototipos electrónicos (arduino, Jetson Nano). En la actualidad, se cuenta con los recursos tecnológicos necesarios que permiten la implementación de nuevas herramientas que facilitan el procesamiento y la geolocalización en tiempo real, ya que es uno de los ejes principales del enfoque del área de Seguridad Ciudadana e integración de las tecnologías de la información y comunicación en protección y garantía en la integridad de la sociedad, siendo este un proceso a ser gestionado con prioridad, ya que sin seguridad ciudadana no tenemos desarrollo humano ni protección que garantice nuestra calidad de vida.

Con el uso de las TIC's podemos disminuir la inseguridad, sobre todo, cuando se trata de localizar a una persona que se encuentra prófuga de la justicia o desaparecido dentro del territorio nacional. Por todo lo antes mencionado, esta investigación aplicada pretende suplir una necesidad de la sociedad y ayudar a las autoridades en su labor de búsqueda, extendiendo el uso de estas tecnologías adaptables y/o herramientas que soporten y motiven al desarrollo de estas áreas, en los medios de transporte públicos. El presente sistema posee un repositorio de individuos a ser rastreados, el cual se implementa usando un prototipo en medios de transportes públicos, que permitirá su detección, por lo que se pretende dar una alternativa y mejor herramienta para reconocer a los prófugos de la justicia y encontrar a personas desaparecidas, con el objetivo de brindar mayor seguridad a los ciudadanos y la optimización en los tiempos de respuesta.

El desarrollo del aplicativo está compuesto de una aplicación móvil, donde la ciudadanía podrá interactuar para formular las siguientes solicitudes: denuncia de tipo de casos, notificaciones, información de casos, módulo wantedface para detectar un posible caso desde la app. En la aplicación web esta tiene 3 principales funciones: mantenimiento general, administración cámaras, a través de la cual se pueden gestionar las diferentes áreas, que se encuentra dentro del sistema como la sala de cámaras, donde se puede seleccionar el tipo de cámara para hacer un seguimiento minucioso.

4. TRABAJOS RELACIONADOS

En el presente proyecto de los autores Garcés Nuñez, Jurado Marco, desarrollado en el año 2017 de un "**Sistema de reconocimiento facial con visión artificial para apoyar al ECU 911 con la identificación de personas en la lista de los más buscados**" [1], estos autores estudian la implementación de un prototipo en tiempo real de la identificación de personas buscados por la justicia que se llevará en la zona 3 del ECU 911, ya que estos están en la lista de los más buscados, el objetivo primordial es de incrementar la seguridad en la ciudadanía, brindando un sistema que será usado en varios lugares, los cuales pueden ser medios de, transporte público, etc., en beneficio de la población.

El autor Villalon de la Vega, D. E implementa un "**Diseño e implementación de una plataforma de software para reconocimiento facial en video**" [2], hoy en día van creciendo los avances en reconocimiento de rostro de video que son correspondientes a las iniciativas privadas y comunidades abiertas, planteando al diseño, implementación de una plataforma válida de software para el reconocimiento mediante archivo de video.

Mariño, C., y Lewis, denomina una, "**Aplicación móvil de reconocimiento facial en personas con antecedentes de abuso sexual en la provincia de Andahuaylas, Apurímac-2018. 2018**." [3], este ayuda a la justicia en la búsqueda de las personas más buscadas en el delito de abuso sexual, este será una aplicación de reconocimiento facial destinado a la identificación y reconocimiento de personas en tiempo real, llevado a cabo en la dirección zonal de Andahuaylas, para ser aplicado a la localización de individuos que se encuentran en una lista denominada.

Auquilla Cayambe, M. A., Cuásquer, A., y Andrés, J. Crean un "**Desarrollo de un prototipo de una aplicación móvil híbrida de reconocimiento facial para la DINASED de la ciudad de Riobamba**" [4]. El pretende suplir a la seguridad de la ciudadana "de una aplicación móvil híbrida de reconocimiento facial, se utilizó Android Studio y la librería Opencv que permitieron la implementación del algoritmo de reconocimiento facial"

Garces Guayta Lucas Rogerio, Alvarado Salinas Yasser Cesar, Paladines Enriquez Nixon Rafael en su proyecto de "Procesamiento de video utilizando Apache Hadoop con

OpenCV y JavaCV para reconocimiento facial" [5], realizaron un análisis para reconocimiento facial de videos, está alojado en una base de datos, permitiendo a las cámaras del ECU-911 y las que se encuentran instalados en los establecimientos públicos, privados dentro del país para dar un mejoramiento en la seguridad.

Los autores Velarde Vargas Olenka Ingrid, Yabarrena Tamayo Jose Samuel en su tesis "**Sistema de reconocimiento facial para el control de la trata de personas en Perú**" [6], fue implementado en Perú, el cual está contribuido al control de la trata de personas en Perú.

Niola Quito Christian Xavier, Zhinin Sanango Armando Willam en su proyecto de titulación "**Desarrollo de un software de seguridad para detección y reconocimiento facial basado en los algoritmos de Viola-Jones y PCA Eigenface**" [7], este proyecto está basado para detección y reconocimiento facial con condiciones controladas de una solución biométrica, para identificación de las personas.

Olvera Olbera Karen Angeline en proyecto de titulación "**Diseño de un sistema que ayude en la búsqueda de personas desaparecidas mediante reconocimiento facial. 2018**" [8], la propuesta es basado a la problemática que tiene el país, su objetivo principal de este proyecto ayudar en la búsqueda de las personas desapercibidas por medio de reconocimiento facial.

5. MARCO TEORICO

En este capítulo trataremos de hablar de las tecnologías que son aplicadas en desarrollo de la tesis el tratamiento de cada uno de estos temas, se lo realiza de una forma superficial ya que son componentes, sin embargo, son el eje central de esta tesis.

Los temas abordados son los siguientes.

5.1 Fog Computing

Fog Computing es una plataforma altamente virtualizada que proporciona servicios de computación, almacenamiento y redes entre dispositivos finales y centros de datos de computación en la nube tradicional, típicamente, pero no exclusivamente, ubicados en el borde de la red [1]. Su objetivo principal es reducir la transferencia de datos dentro de la comunicación entre los dispositivos finales y la nube en las redes externas, donde

se ve qué datos se procesarán localmente y cuáles serán enviados a la nube para luego ser analizados o procesados.

Debido a la documentación técnica de OpenFog, cómo el estándar oficial para Fog Computing, ofrece varias ventajas respecto a otros enfoques, los cuales se rigen bajo el término “SCALE” **[29]**.

- ✓ Seguridad: brinda seguridad adicional permitiendo garantizar las transacciones seguras y confiables.
- ✓ Conocimiento: permite conocer los objetivos y los requerimientos del cliente para tener control desde la nube hasta los dispositivos.
- ✓ Agilidad: permite una innovación rápida y escalabilidad factible para la infraestructura; los clientes pueden usar interfaces de programación y kits para el desarrollo de nuevos servicios.
- ✓ Latencia: permite procesar en tiempo real como monitoreo médico, domótica, etc.
- ✓ Eficiencia: permite agrupar los recursos locales (nodos) que se encuentran distribuidos geográficamente.

Fig. 1. Pilares de la arquitectura de Fog Computing [29]

La arquitectura de Fog Computing [29] cuenta con ciertos alineamientos para su implementación, a continuación, se describirán cada uno de ellos.

- ✓ Seguridad: muchas aplicaciones de IoT compatible con Fog Computing sufren de aspectos de privacidad, misión crítica, tiempos de vida bajos; la implementación de seguridad se basa en atributos como privacidad, anonimato, integridad, confianza, certificación, verificación y medición; se necesita descubrir, atestiguar y verificar las "cosas inteligentes" antes de poder establecer una confianza **[29].**
- ✓ Escalabilidad: las propiedades de Fog Computing y la ubicación de los diferentes nodos por las redes simples y complejas agregan la oportunidad de poder escalar mediante la adición de hardware y software, adición de nodos en la niebla bajo demanda, servicios de almacenamiento permiten escalar dentro de la infraestructura de niebla **[29]**.
- ✓ Sistema abierto: al tener un sistema abierto es esencial para el éxito en la implementación de una arquitectura Fog Computing, permitiendo añadir nuevos nodos y que sean descubiertos automáticamente para que puedan existir en cualquier lugar de una red **[29]**.
- ✓ Autonomía: en presencia de fallas externas, en ciertas veces ante la inesperada indisponibilidad en los centros de datos, los servicios y nodos deben trabajar sin ninguna interrupción, deben seguir otorgando servicios a los usuarios finales, podemos concluir que los nodos deben satisfacer las peticiones de los usuarios aun cuando la nube pueda estar inoperativa **[29]**.
- ✓ Programabilidad: permite implementaciones adaptables donde incluyen soporte para la programación tanto para el software y hardware, es decir podemos reasignar tareas a uno nodo o grupo de nodos dinámicamente y automatizado **[29].**
- ✓ Agilidad: los usuarios no tienen la capacidad de analizar la cantidad de datos que generan los sistemas IoT donde los datos son analizados y procesados en servidores dedicados en la nube tardando desde la

solicitud hasta la respuesta; ante este inconveniente es necesario que la capacidad de análisis de los datos y el procesamiento de los mismos sea lo más cercano al origen de la solicitud **[29].**

- ✓ Jerarquización: depende de la escala y naturaleza del escenario antes de la implementación desde una red de sistemas particionados hasta un solo sistema físico **[29].**

Este modelo o paradigma permite sacar provecho a las aplicaciones informáticas inteligentes expandiendo sus funcionalidades no desde un servidor lejano sino desde su ubicación geográfica o desde los mismos dispositivos, en lugar de realizar los cálculos complejos solicitando a un servidor remoto podemos explotar tales cálculos o procesos en pequeños micro servicios empaquetados en contenedores docker [2], en conclusión juntando todas estas tecnologías, se puede tener mayor velocidad en desarrollo, implementación, y control de cambios como su reconfiguración.

El objetivo es abordar una arquitectura sostenible a QoS que admita aplicaciones inteligentes bajo una numerosa carga de trabajo para los dispositivos, que soporten capacidades computacionales requeridas por IA generados por tales dispositivos.

La ilustración en la (fig., 2) presenta la arquitectura que define la información y computación que soporta Fog Computing.

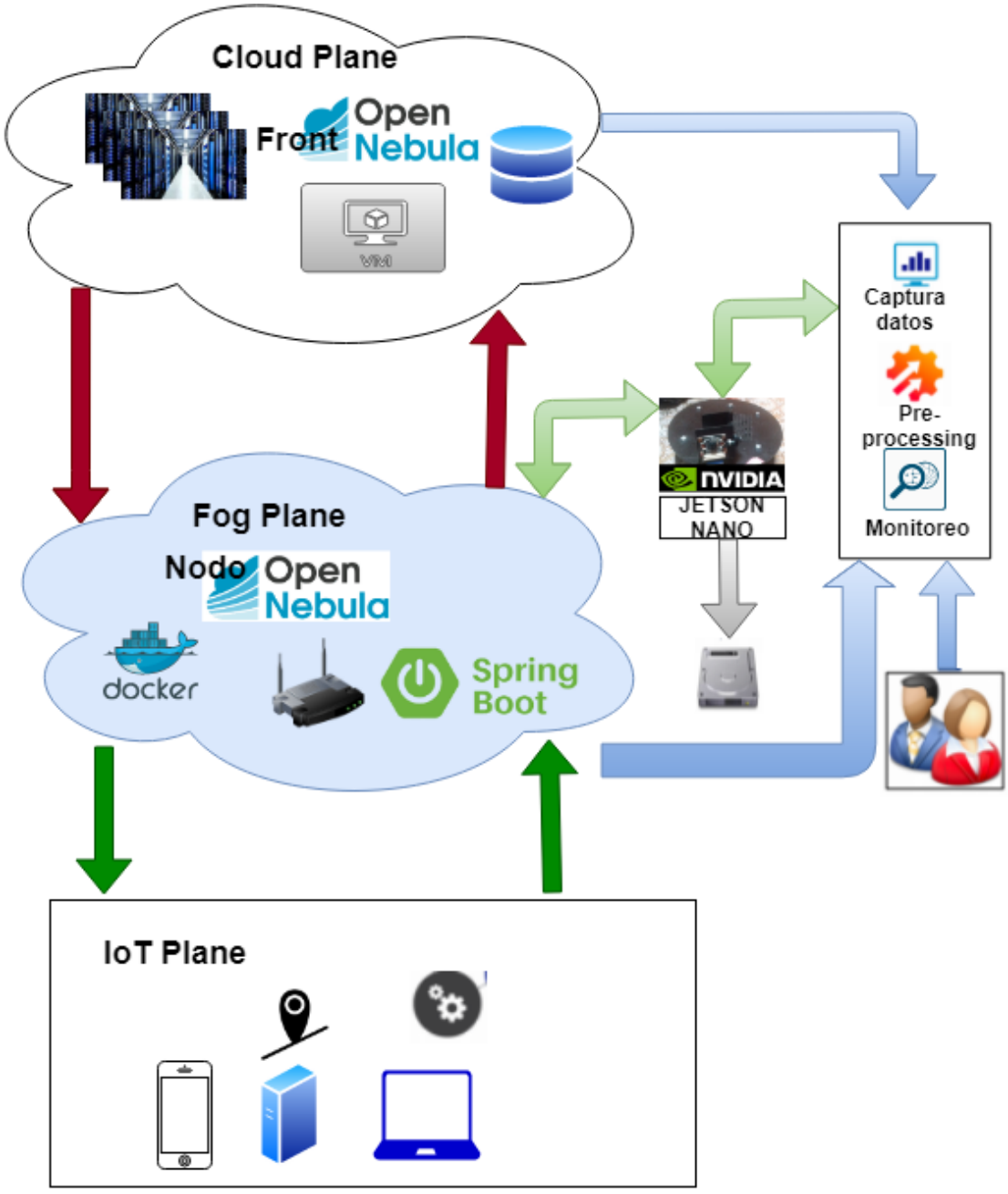


Fig. 2. IoT & Fog Computing.
Fuente: Elaborado por los Autores

5.2 Cloud Computing

Según expone [17], es un modelo que permite el acceso al mismo tiempo desde cualquier parte de la red bajo demanda a un conjunto compartido de recursos informáticos que pueden ser configurables y monitorizados, éstas pueden ser redes, servidores, repositorios de datos o almacenamiento, aplicaciones y servicios que pueden aprovisionarse y liberarse rápidamente y sin esfuerzo de administración; los

clientes pueden interactuar con los servicios en la nube, ya que ofrece los servicios a nivel de software podemos utilizar navegadores, dispositivos móviles.

El elemento principal dentro de Cloud Computing es la virtualización, donde la infraestructura (redes, firewall, almacenamiento, nodos o servidores) se las manipula a nivel de software beneficiando al área TI, en costos, administración, tiempo; toda esta manipulación y configuración es la resultante de Cloud Computing

Según [31], existen algunas ventajas respecto a implementaciones tradicionales como la agilidad, menor costo de implementación, dispositivos independientes, independencia de ubicación y escalabilidad.

Cloud Computing en los últimos años es un nuevo paradigma de cálculo o escenario de las plataformas TI [18].

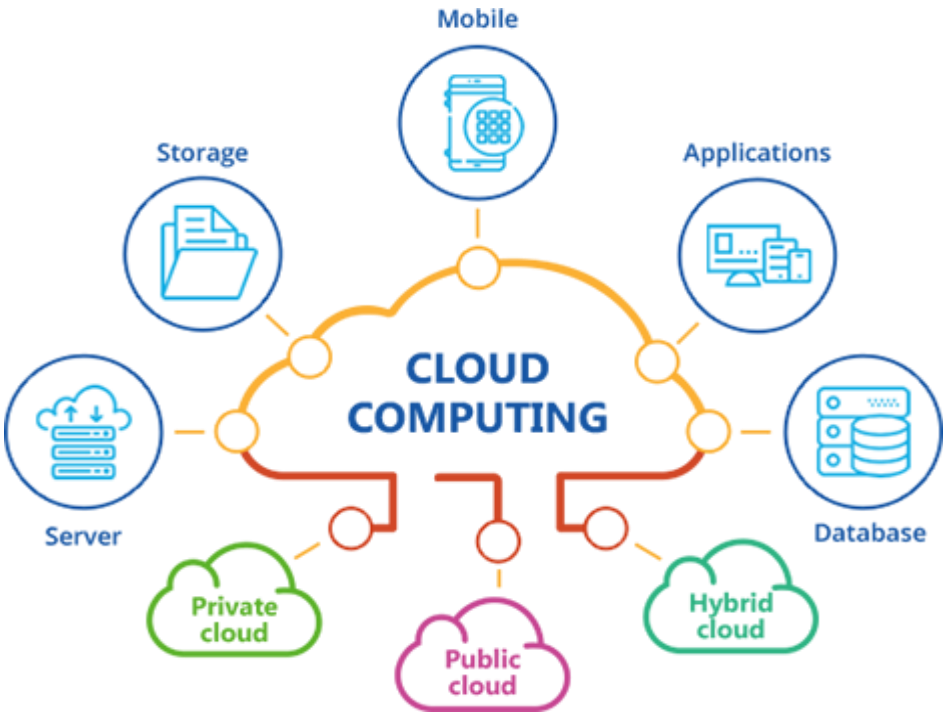


Fig. 3. Cloud Computing [28]

5.2.1 Tipos de Cloud Computing

Según [14] [15] existen tres tipos de nubes que son:

- ✓ **Nubes públicas:** este tipo de nube se implementa para un conjunto de empresas u organizaciones, utilizando un único conjunto de recursos que son administrados por un tercero [15] [16].

- ✓ **Nubes Privadas:** en este tipo de nube, se aplica de forma privada, es decir, la plataforma se encuentra dentro las instalaciones de la empresa y sólo esa organización tiene acceso a los recursos [15] [16].
- ✓ **Nubes Hibridas:** combinan recursos locales de una nube privada y las de la nube pública. Este tipo de nube es de utilidad cuando una empresa u organización requiere aumentar sus recursos privados dentro de la demanda de determinados servicios, por lo cual la empresa contrata servicios de Cloud público para la expendición de los recursos del Cloud privado.

5.2.2 Capas de Cloud Computing

- ✓ **Software como Servicio (SaaS):** son aplicaciones de software adecuados en internet y se las puede utilizar sin necesidad de ser instaladas locamente en una computadora., consiste en la entrega de aplicaciones completas como servicio a través de Internet, los proveedores de este servicio tienen instalada la aplicación en sus servidores propios" [16].
- ✓ **Plataforma como Servicio (PaaS):** dentro de esta plataforma permite virtualizar los servidores de una empresa para proveer todas las necesidades de las aplicaciones desarrolladas, esto significa que contratan proveedores que ofrecen máquinas virtuales en sus centros de cálculo" [16].
- ✓ **Infraestructura como Servicio (IaaS):** en esta infraestructura se representa a las grandes capacidades de almacenamiento masivo de información, los clientes compran los recursos como un servicio externo. Los proveedores cobran los servicios según la base establecida y por la cantidad de recursos consumidos [16].

5.3 Docker

Docker es un proyecto "open source", "Permite crear aplicaciones en contenedores de software que son ligeros, portátiles y autosuficientes" [7] dentro del software de los contenedores el cual contiene una capa adicional de abstracción y automatización de virtualización de aplicaciones en diferentes sistemas operativos. "Es una plataforma donde los usuarios pueden empaquetar, distribuir y administrar aplicaciones dentro de contenedores" [8].

Permite empaquetar procesos complejos y servicios en un contenedor, resultando un paquete listo para desplegar en algún dispositivo, además que no necesita un sistema operativo para funcionar independiente del sistema operativo que lo reside.

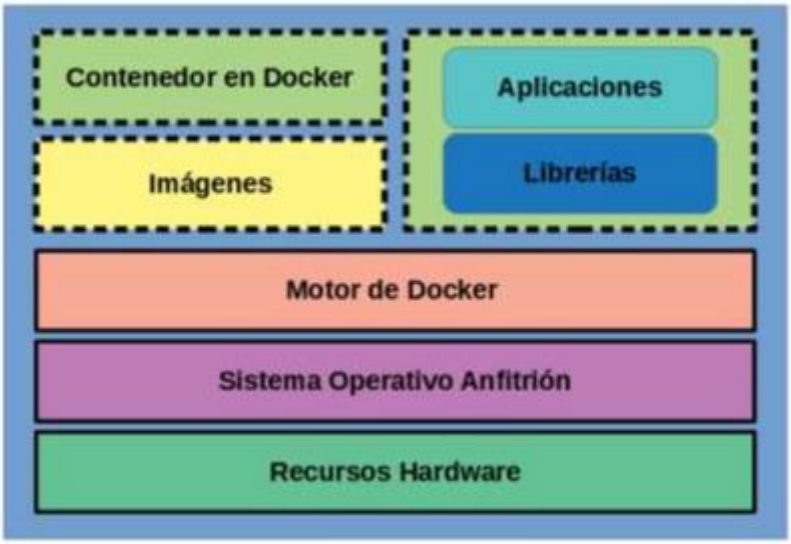


Fig. 4. Infraestructura de Contenedor [37]

5.3.1 Características

Las características de docker son [18]:

- ✓ **Portabilidad:** el contenedor Docker podemos desplegarlo en cualquier sistema, sin necesidad de volver a configurarlo o realizar las instalaciones necesarias para que la aplicación funcione, ya que todas las dependencias son empaquetadas con la aplicación en el contenedor.
- ✓ **Ligereza:** los contenedores Docker sólo contienen lo que la diferencia del sistema operativo en el que se ejecutan, puede o no virtualizar un SO completo.
- ✓ **Autosuficiencia:** un contenedor Docker no contiene todo un sistema operativo completo, sólo aquellas librerías, archivos y configuraciones necesarias para desplegar las funcionalidades que contenga.

5.4 Arquitectura microservicios

En lugar de ofrecer una arquitectura monolítica, donde sus elementos (servicios, procesos) se encuentran alojados y compilados en el mismo lugar o centralizado a la vez y donde la capa de negocio suministra y acoge a toda la capa de presentación, es decir toda la carga proveniente del cliente se ejecuta desde un solo proceso, ya que está en un solo proyecto grande [3]; podemos segregar servicios o procesos bien

probados conocidos como componentes de software en microservicios donde cada servicios es responsable de su propio objetivo.

La desventaja es ya sea un pequeño cambio en el código, en el sistema monolítico se necesitará compilar y luego desplegar el cambio; en la de microservicios cada capacidad es un servicio independiente y autónomo, cabe señalar que cada microservicio que se ejecute en un dispositivo agilita la administración en tiempo de ejecución que incluye operaciones como implementación, creación de instancias, etc.

La (fig., 5) compara la arquitectura monolítica con la de microservicios.

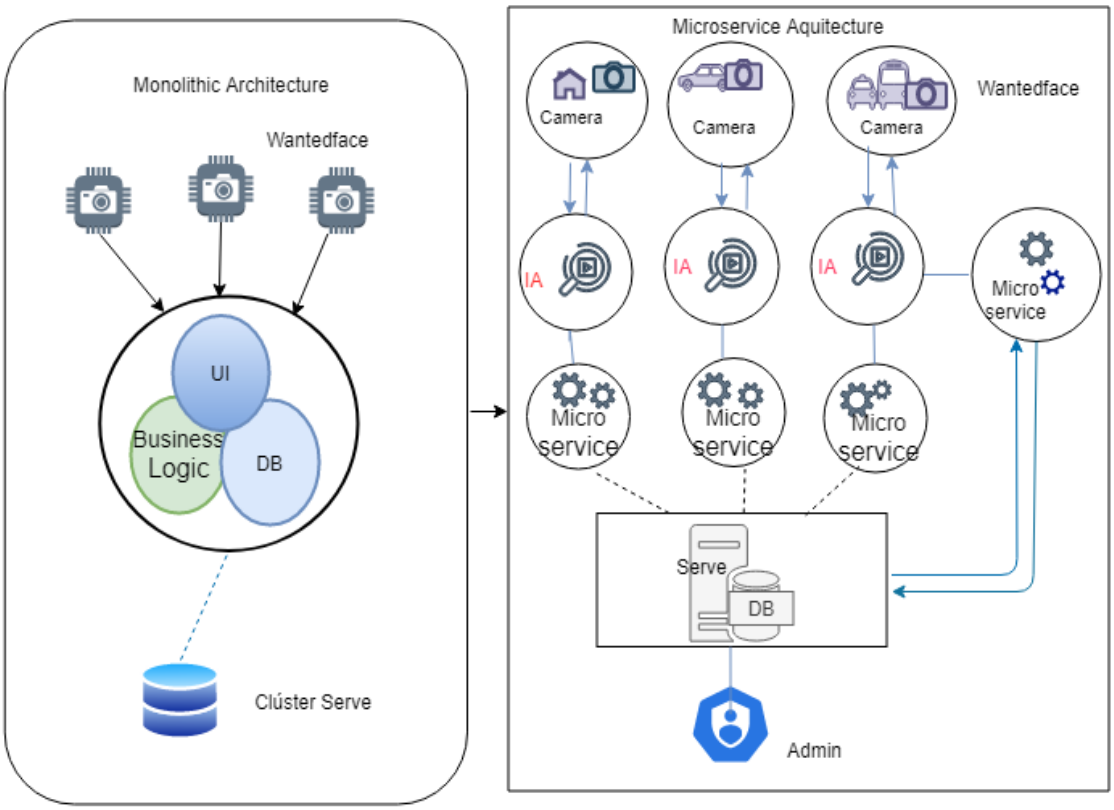


Fig. 5. Arquitectura Microservicios

Fuente: Elaborado por los Autores

5.4.1 Microservicios

"Pequeños servicios autónomos que trabajan juntos" [4]. Es un enfoque para el desarrollo de una aplicación única como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y mecanismos ligeros de comunicación, a menudo un recurso de una interfaz de programación de aplicaciones (API) sobre protocolo de transferencia de hipertexto (HTTP). Estos servicios están construidos alrededor de las

capacidades del negocio y con independencia de despliegue e implementación totalmente automatizada.

Existe un mínimo de gestión centralizada de estos servicios, los que pueden estar escritos en lenguajes de programación diferentes y utilizar diferentes tecnologías datos de almacenamiento [5].

Podemos definir microservicios bajos estos criterios:

- ✓ **Enfocado en la facultad del negocio:** el desarrollo es más manejable con microservicios, ya que cada microservicio tiene su propia lógica de negocio, conexión con base de datos, comunicación con otros servicios y tecnologías, resultando que el grupo de desarrollo pueda agregar nuevas propiedades o cambios incluso para testeo bajo demanda sin afectar el funcionamiento ni la estabilidad del sistema.
- ✓ **Independencia de servicios autónomos:** cada servicio tiene su propia lógica de negocio y se despliega por separado por tener embebidos servidores de aplicaciones pudiendo correr o desplegarse en diferentes puertos, es autónoma ya que permite la flexibilidad de escoger la tecnología más adecuada para un servidor en específico.
- ✓ **Descentralización de base de datos:** todos los servicios apuntan a su propia base de datos en esquemas y pequeños y simples de configuración.
- ✓ **Tolerancia a fallos:** como son sistemas pequeños desplegados en diferentes puntos estos pueden fallar, estas unidades están acopladas y monitoreadas pudiendo ser restauradas automáticamente o bajo demanda.
- ✓ **Interfaces y comunicación:** cada microservicio debe exponer un punto de comunicación, esta puede replicarse a otro microservicio o tener diferentes, la comunicación se la puede representar por REST.
- ✓ **Transacciones y bloqueos:** se debe asegurar que ningún microservicio utilize flujos cerrados o recursos bloqueados y peor cambiar datos que otros servicios lo están usando para eso entra el Gateway ZUUL (3.3).
- ✓ **Alta disponibilidad**: puede existir microservicios principales y de backup para poder siempre tener los servicios levantados además de brindarme información de cuando un servicio este caído y como recuperarlo.

Para el desarrollo del proyecto técnico cada microservicio cuenta con reglas de negocio capaz de compartir información entre distintas tecnologías e independientes permitiendo agregar código o nuevos cambios con testeo sin afectar el funcionamiento de los servicios; además que se publican tales reglas de negocios bajo un puerto conocido y se comunican por http (rest, post, etc) permitiendo interactuar cada microservicio con diferentes base de datos e incluso poder recuperarse en caso de fallas, tener la capacidad de verificar remotamente el servicio y replicarlo.

5.5 Spring Boot

Es un framework Java basado en el Modelo Vista Controlador, mediante el cual gracias a los componentes y librerías que brinda hace fácil el desarrollo y despliegue de los servicios REST. Ha eliminado la necesidad de configurar la aplicación con el uso de archivos XML haciendo énfasis en el desarrollo de la misma [6].

La ventaja es que permite crear aplicaciones independientes ya que puede tener servidores de aplicaciones embebidos como Tomcat, Jetty; además como mencione brinda servicios web RESTful y admite inyección de dependencias por lo que facilita el uso de servicios de terceros el cual facilita el desarrollo, resultando más conveniente.

El punto clave de este framework es la creación de aplicaciones independientes y la inyección de dependencias pudiendo usar de terceros como Netflix el cual favorece a la creación de microservicios.

Spring Boot ofrece entre tantas tecnologías o servicios, los que distinguen son: como tal un microservicio (3.4), API Gateway el cual abreviaremos adelante como enrutador, Service Discovery el cual abreviaremos como diccionario.

API Gateway: en microservicios es una arquitectura, el cual es un elemento necesario que sirve para las distintas solicitudes de los clientes e interfaces del usuario como navegadores web, aplicaciones móviles, dispositivos electrónicos, etc., ya que cada cliente tiene sus necesidades que incluso puede cambiar con el paso del tiempo, entonces API Gateway recepta solicitudes de clientes de distinta naturaleza o tecnología, proporciona funcionalidades para publicar múltiples API's cada uno dedicado a un grupo de clientes.

Service Discovery: en la práctica, la ubicación de un microservicio puede no conocerse estáticamente en el momento del diseño, esto es debido a que los microservicios

pueden implementarse en un sistema que está basado en la nube independiente, donde se puede desplegar, replicar uno ya existente o simplemente reubicarlo recordando que pueden correr bajo un servidor de aplicaciones publicado bajo un puerto, esto exige usar algún mecanismo de descubridor de servicio bajo un registro de servicio, donde los microservicios puedan hablar con el registro para informar de su ubicación.

5.6 Vision Artificial

"Es una disciplina que engloba todos los procesos y elementos que proporcionan "ojos" a una máquina y se podría decir que: la visión artificial o comprensión de imágenes describe la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, bien a partir de una o varias imágenes bidimensionales de ese mundo" [19] también exponen que es "un sistema autónomo que realiza alguna de las tareas que el sistema de visión humano es capaz de realizar" [20], también permite "construir nuevos y más sofisticados algoritmos que sean capaces de obtener información de bajo nivel visual; y aunque todavía se está años luz de la percepción visual de los seres vivos, la Visión Artificial es muy eficaz en tareas visuales repetitivas y alienantes para el hombre" [21].

"La visón artificial o visión por computador" [23] reproduce artificialmente el sentido de la vista humana mediante el procesamiento e interpretación de imágenes que son captadas desde diferentes dispositivos como pueden ser cámaras, las cuales pueden utilizar computadores para su procesamiento:

El sistema visión artificial [23] tiene dos elementos primordiales que son:

- ✓ Hardware que se encarga de la percepción de las imágenes
- ✓ Software que se encarga del procesamiento de la información.

En vista a la complejidad del proceso, la cantidad de información se va clasificando hasta obtener objeto buscado en cual se divide en las siguientes etapas o faces que son [23]:

- ✓ **Captura:** realiza la captura de imágenes por medio de un sensor

- ✓ **Análisis de (tratamiento digital):** en la aplicación de filtros se separan partes de la imagen donde se enfoca la probabilidad con mayor coincidencia en lo buscado.
- ✓ **Segmentación:** consiste en aislar los elementos de la imagen que se desean analizar, La segmentación permite comprender la imagen individualizando sus elementos.
- ✓ **Reconocimiento:** En ella se distinguen los objetos segmentados, mediante al análisis de ciertas características, que se establecen previamente para diferenciarlos.

Al implementar estos procesos mediante algoritmos de procesamiento y se puede identificar el objeto buscado

5.7 Herramientas de software elegidas para el desarrollo

El software que se usó para el desarrollo y testeo del sistema fue elegido en base a su flexibilidad, estabilidad y su rendimiento ante una gran cantidad de trabajo simultáneo en tiempo real.

- **Python 3:** "es un lenguaje de programación de alto nivel y de un propósito general" [28]. Python está formado por varios paradigmas de programación dentro ello contiene gestión de memoria, Su "filosofía la misma que Linux, está gestionado por la organización de software de Python" [9].
- **Ionic:** es un potente Framework para el desarrollo de las aplicaciones móviles, híbridas, permitiendo el desarrollo de las aplicaciones en un reducido periodo de tiempo, el cual es posible compilar de estas diferentes plataformas como es:(IOS, Windows Phone, Android) haciendo uso del mismo código fuente [10]

  Es usado para el desarrollo de la aplicación móvil haciendo uso de este se compiló para plataforma de Android
- **Postman:** "es una herramienta que envía solicitudes a una API y le muestra la respuesta" [7], a través de los "métodos request como: GET, POST, PUT, DELETE, a través de la URL" [11]. Es usado para verificación del funcionamiento de los webs services.

- PostgreSQL: sistema de Base de Datos relacional que está orientado a objetos, que a su vez usa lenguaje de SQL, su arquitectura es confiable e integridad de datos y código abierto [12]. Esta base de datos es usada para el almacenamiento de la información, el cual es registrado desde la aplicación móvil.
- OpenCv: es una biblioteca libre de visón artificial, esto está bajo la licencia BSD y, mediante el cual es gratuito para el uso académico y comercial. Tiene interfaces C++, Java, Python y estos a su vez son compatibles en los sistemas operativos de Windows, Linux, MacOS, Android y IOS. Este fue diseñado para la eficiencia computacional y está enfocado a las aplicaciones en tiempo real. Escrita en C/C++ optimizado, el procesamiento multi-core de la biblioteca se puede aprovechar. Habilitado con OpenCL, se puede aprovechar la aceleración de hardware de la plataforma de cómputo heterogénea subyacente [13].
- Paquetes Adicionales
    - Sockets cliente: es una Librería que permite realizar sockets
    - url lib.request: es una interfaz de programación para usar los recursos de
    - internet identificados por URLs. para el manejo de autenticación de HTTP
    - face_recognition: librería que permite hacer el recogimiento facial
    - numpy: es una extensión de python para vectores y matrices.
    - Base64: procesamiento de imágenes a texto.

6. MARCO METODOLOGICO

En este capítulo hablaremos de la metodología SCRUM aplicada para el desarrollo de esta tesis en las etapas de gestión, desarrollo y en el control de cada fase durante la elaboración del proyecto técnico de reconocimiento facial, desde su inicio del desarrollo hasta el período de culminación.

6.1 Metodología SCRUM

La metodología SCRUM, es una metodología ágil, que está enfocada para gestión de los proyectos de software, que permite la planificación de los proyectos en bloques pequeños o sprint; es decir, que cada proceso que se va avanzando en la fase del proyecto se va haciendo correcciones y mejorando mediante versiones [24] [25]. Esta metodología ayuda a la motivación e innovación y el compromiso del trabajo en equipo que están durante la fase proyecto, además de competir en las diferentes actividades [26].

Sprint: recibe el nombre de cada uno de las iteraciones o ciclos que se tiene dentro del proyecto SCRUM, este puede ser representado como la creación de un mini-proyecto dentro de un proyecto grande con un intervalo determinado de tiempo (a lo que se refiere no tiene que ser inferior a una semana o superior a un mes). Hay sprint que tiene procesos de desarrollo de construcción, estos pueden ser consecutivos o tienen una duración constante.

Etapas de SCRUM [27] las cuales son sus tres grandes etapas:

- ✓ **Inicio**: en esta etapa se conforma por dos sube tapas:
  - Planeamiento: Fundamenta en la construcción de la visión, financiamiento del producto, y ayuda a especificar cuál es la funcionalidad especifica
  - Arquitectura: Análisis y conceptualización del proyecto
- ✓ **Desarrollo**: mediante esta etapa se procede el desarrollo del proyecto, estas se dividen en iteraciones la entrega de los resultados funcionales.

- ✓ **Cierre**: en esta etapa se implementa la documentación, testing y la ejecución de dicho proyecto.

Dentro del proyecto técnico se utilizó la metodología SCRUM en las siguientes etapas que tiene esta metodología, dentro de la etapa de inicio es donde se pudo ver la vialidad del proyecto técnico y en la segunda etapa donde se procedió realizar las fases y procesos para la construcción de cada iteración de la entrega de cada fase y los resultados de cada uno de ellos, el proceso de culminación en la etapa de cierre es donde se obtuvo la entrega y testeo del proyecto técnico de reconocimiento facial.

7. LEVANTAMIENTO DE LOS REQUERIMIENTOS

7.1 Diagramas de casos de uso

Este modelo de caso de uso permite construir, documentar, representar el flujo de comunicación de cada ente o modelo con los respectivos servicios que ofrece el sistema ya sea este aplicativo móvil, dispositivo electrónico o un servidor dedicado embebido de un core transaccional (sistema principal) representado por símbolos que pretende ser fácil de entender al lector.

Actividades de Aplicativo Web:

- ✓ Registro de usuario
- ✓ Página Principal
- ✓ DashBoard
  - ➢ Detectar e identificar el involucrado o sala de seguimiento de rostros
- ✓ Administración
  - ➢ Cámara
  - ➢ Fotos
  - ➢ Notificaciones
- ✓ Crud
  - ➢ Mantenimiento Cámara
  - ➢ Lista casos (Editar, Eliminar)

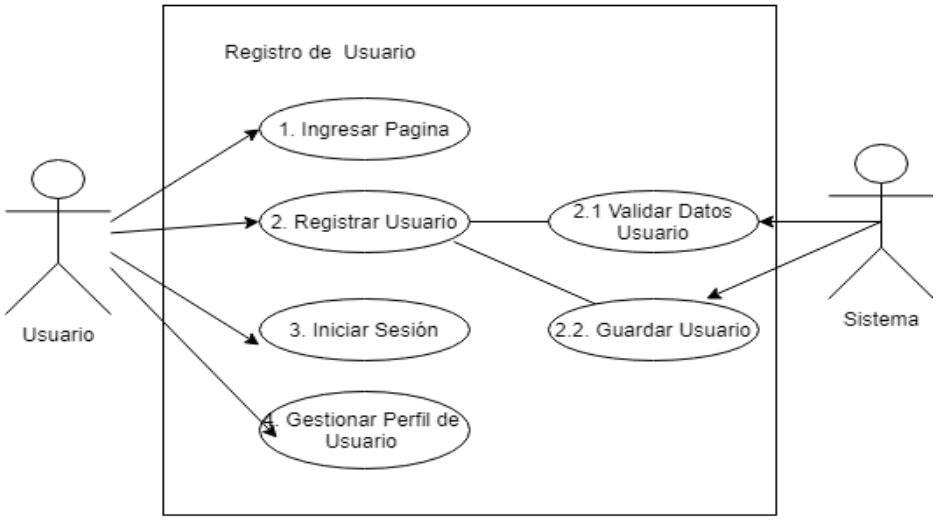


Fig. 6. Diagrama de inicio del sistema web

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso a la aplicación web
Actor	Usuarios, Administración.
Descripción	Acceder al Sistema web
Precondiciones	Abrir página de administración inicial del sistema
Flujo Normal	1. Se ingresa a la pantalla principal de administración web 2. Formulario para el registro de usuario. 2.1 ubicación de los datos del usuario antes de guardarlo 2.2 Al finalizar la validación se guarda en el sistema 3. Inicio de Sesión del usuario 4. Gestionar tipo de perfil para sesión del usuario

Tabla 1. Acceso a la aplicación web

Para el inicio de la aplicación web se puede visualizar la parte navegable de las páginas, para el registro e información de los usuarios de administrador que facilitaran el acceso a los mismo tal como se puede ver en la (fig. 6) que lo representa.

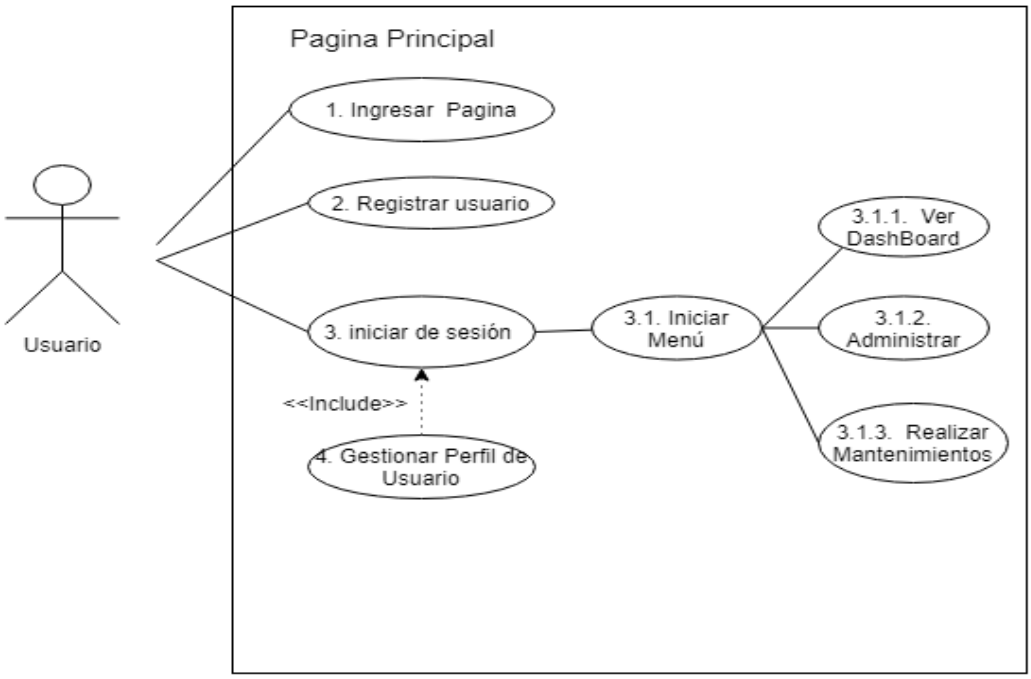


Fig. 7. Diagrama de acceso al menú inicio

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso al Sistema de administración web
Actor	Usuarios, Administración
Descripción	Acceder al Sistema.
Precondiciones	Ser aceptado por el administrador como root con su respectivo usuario y contraseña
Flujo Normal	1. Se ingresa a la pantalla principal de administración web 2. Formulario para el registro de usuario. 3. Inicio de Sesión del usuario 3.1 Inicia el menú principal del Dashboard del usuario logueado. 3.1.1 Dashboard principal de ubicación de cámaras y casos de uso. 3.1.2 Pagina de seguimiento de cámaras en tiempo real y despliegue de la meta data 3.1.3 Mantenimiento de cámaras

Tabla 2. Acceso al sistema de administración web

Una vez realizado correctamente el inicio de sesión para la administración se puede

visualizar el menú correspondiente a la página principal para los diferentes módulos implementados en la aplicación como se indica en la (fig.7)

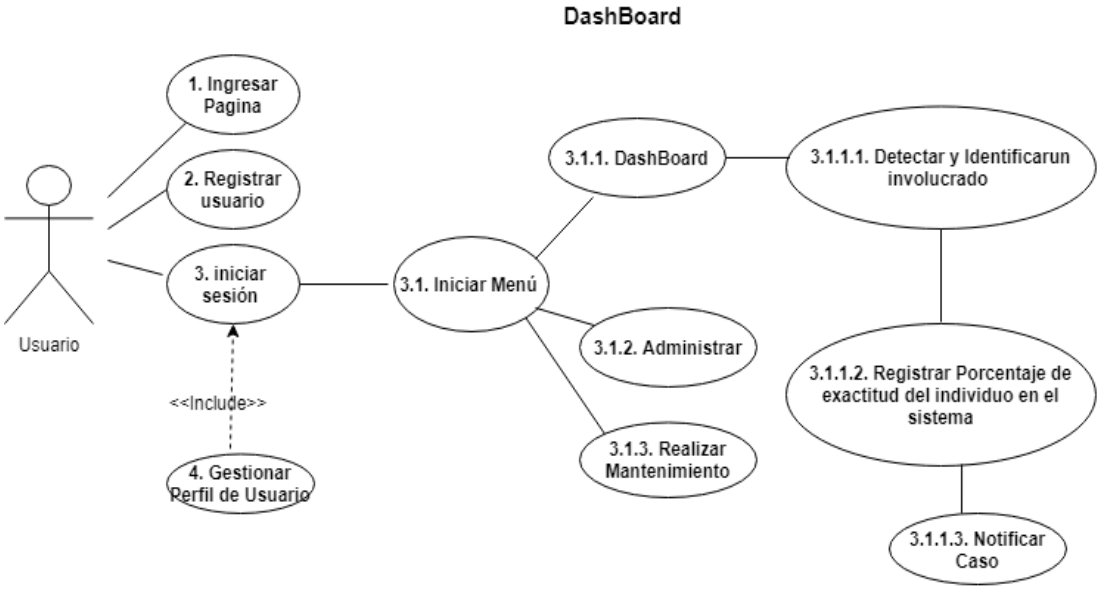


Fig. 8. Diagrama de acceso al módulo de control DashBoard.

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso al sistema de control (DashBoard)
Actor	Usuarios, Administración
Descripción	Acceder al Sistema desde el menú principal
Precondiciones	Ser aceptado por el administrador como root y acceder al módulo desde el menú
Flujo Normal	1. Se ingresa desde el menú de navegación de la aplicación 2. Se puede visualizar los porcentajes de cada individuo en el registro 3. Inicio de Sesión del usuario 3.1 Inicia el menú principal del Dashboard del usuario logueo 3.1.1.1 Modulo que detecta e identifica en tiempo real, en las distintas cámaras 3.1.1.2 Identificado el usuario, se muestra el porcentaje de exactitud o parentesco con la foto original 3.1.1.3. Notificar caso mediante emails

Tabla 3. Acceso al sistema de control DasBoard

En esta parte aplicativa se podrá reconocer mediante una prueba rápida el uso del sistema y su exactitud al reconocer rostros registrados en la base de datos y constatar su precisión al momento de identificarlos en cámara. (fig., 8)

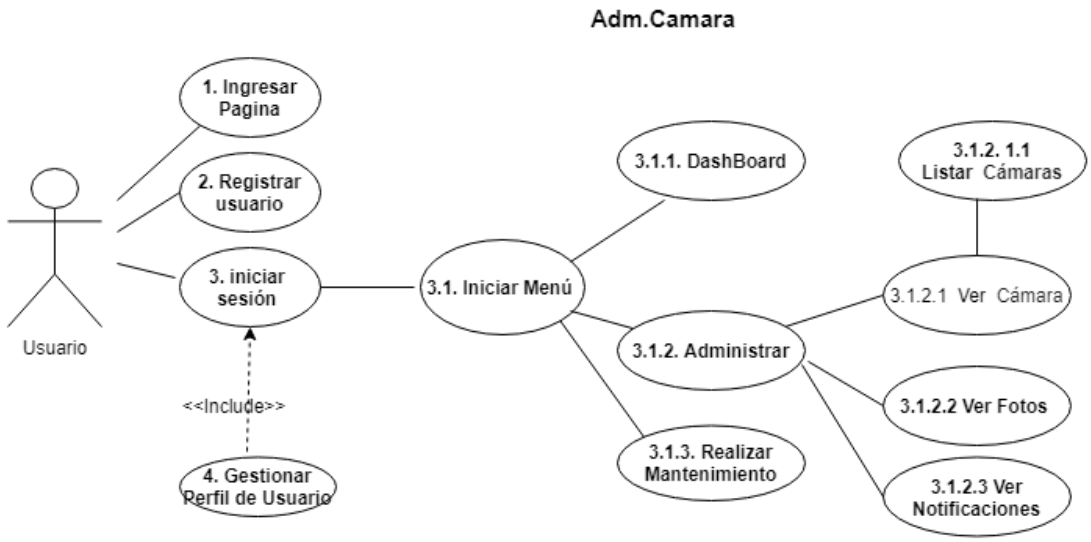


Fig. 9. Diagrama de acceso al sistema de control administrativo por cámara

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso al Sistema de control administrativo por cámara
Actor	Usuarios, Administración
Descripción	Acceder al Sistema
Precondiciones	Ser aceptado por el administrador como root y acceder desde el menú a la opción de administración por cámara
Flujo Normal	1. 3.1.2.1 Muestra las distintas cámaras desplegadas 2. 3.1.2.2 Muestra todas las fotos de los distintos casos 3. 3.1.2.3 En lista notificaciones de los rostros detectados 3.1.21.1 En lista las cámaras.

Tabla 4. Acceso al sistema de control administrativo por cámara

En esta sección podremos visualizar el control de las diferentes cámaras que se encuentran instaladas en el sistema a las que tenemos acceso y su respectiva

información. (fig., 9)

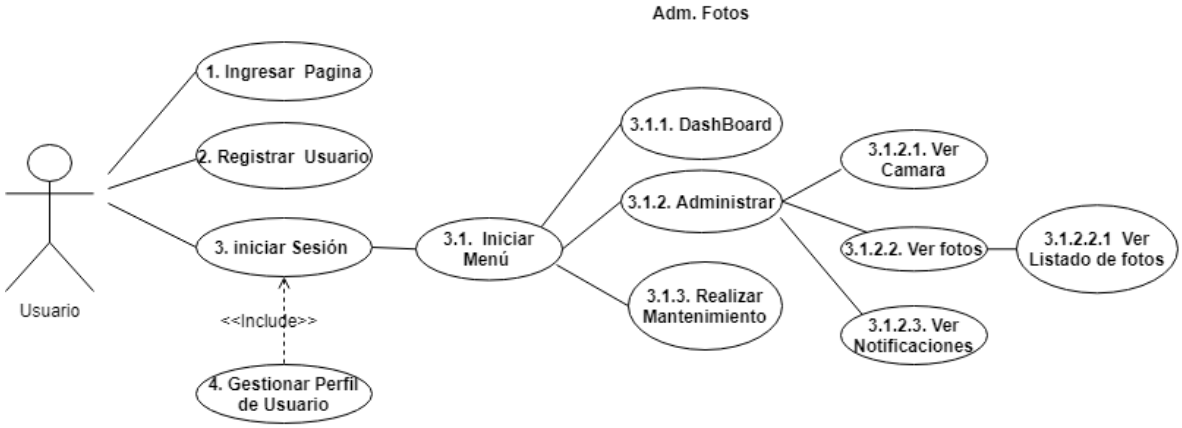


Fig. 10. Diagrama acceso al sistema de control administrativo por fotos.

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso al Sistema de control administrativo por fotos
Actor	Usuarios, Administración
Descripción	Acceder al Sistema de control de fotos
Precondiciones	Ser aceptado por el administrador como root. ingresar al módulo desde el menú principal del sistema
Flujo Normal	1. 3.1.2.2 Se ingresa al módulo desde el menú de la aplicación 3.1.2.2.1 Ver las fotos de los diferentes casos detalladamente.

Tabla 5. Acceso al sistema de control administrativo por fotos

En esta sección del módulo se puede visualizar las fotos desplegadas en las distintas cámaras, además se puede desplegar fotos nuevas en caso de no haber enviado. (fig. 10)

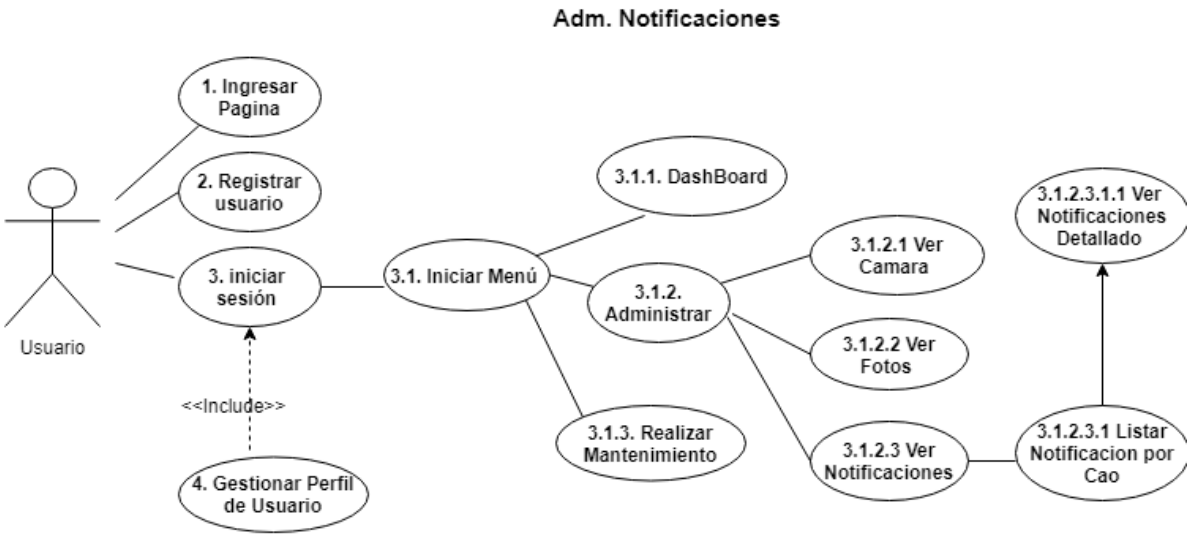


Fig. 11. Diagrama de acceso al sistema de control administrativo por notificados

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso al Sistema de control administrativo por notificaciones
Actor	Usuarios, Admiración
Descripción	Acceder al Sistema de módulo administrativo por notificaciones
Precondiciones	Ser aceptado por el administrador como root, y acceder desde el menú principal al módulo de notificaciones
Flujo Normal	1. Se ingresa al menú principal de la aplicación web 2. Se busca el módulo de administración por notificaciones 3. Se puede visualizar las notificaciones recibidas de los diferentes casos registrados en el listado de control 3.1.2.3 Ver notificaciones agrupado por caso 3.1.2.3.1 En lista las notificaciones de un caso en 3.1.2.3.1.1 Ver información de un caso en una fecha especificada.

Tabla 6. Acceso al sistema de control administrativo por notificaciones

En esta sección del módulo se puede visualizar por fotos un listado de los registros en los que se ha conseguido encontrar o capturar una coincidencia de la persona buscada en una cámara con su respectiva información en lugar y fecha. (fig. 11)

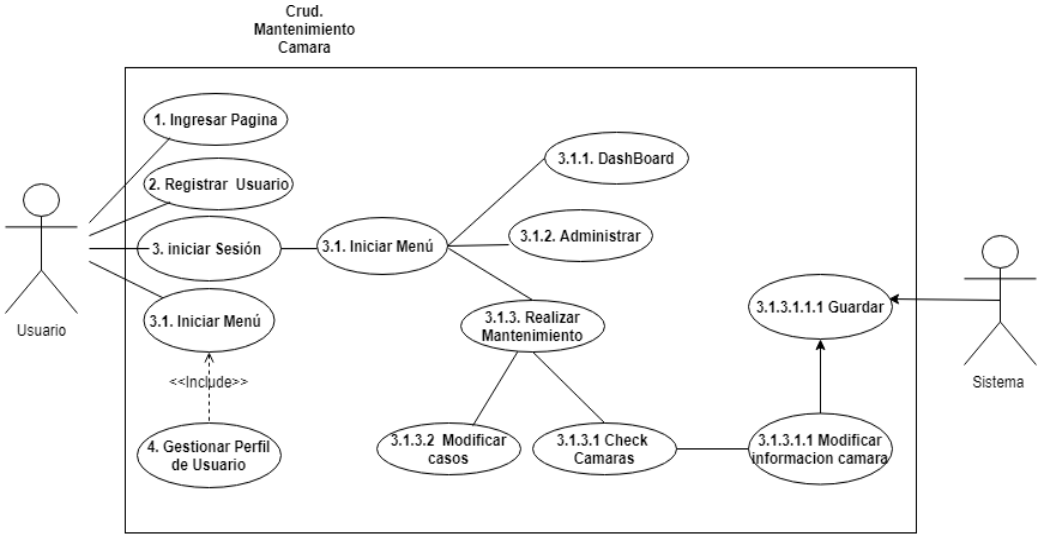


Fig. 12. Diagrama de acceso al sistema de control crud de mantenimiento de cámaras

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso al módulo Crud de mantenimiento de cámaras
Actor	Administración
Descripción	Acceder al Sistema de mantenimiento de cámaras
Precondiciones	Ser aceptado por el administrador como root y acceder al módulo de mantenimiento
Flujo Normal	1. Se ingresa al módulo Mantenimiento del menú de inicio 2. Se selecciona el área de mantenimiento de cámaras 3.1.3 Selecciona el área de mantenimiento general de las cámaras 3.1.3.1 Se realiza mantenimiento de una cámara en específico.

Tabla 7. Acceso al módulo crud de mantenimiento de cámaras

En esta sección se puede manejar el control de los dispositivos instalados para el monitoreo (cámaras) de la aplicación para su respectivo soporte o mantenimiento en accesos y comunicación de estos como la IP de red y marca la ubicación (fig., 12)

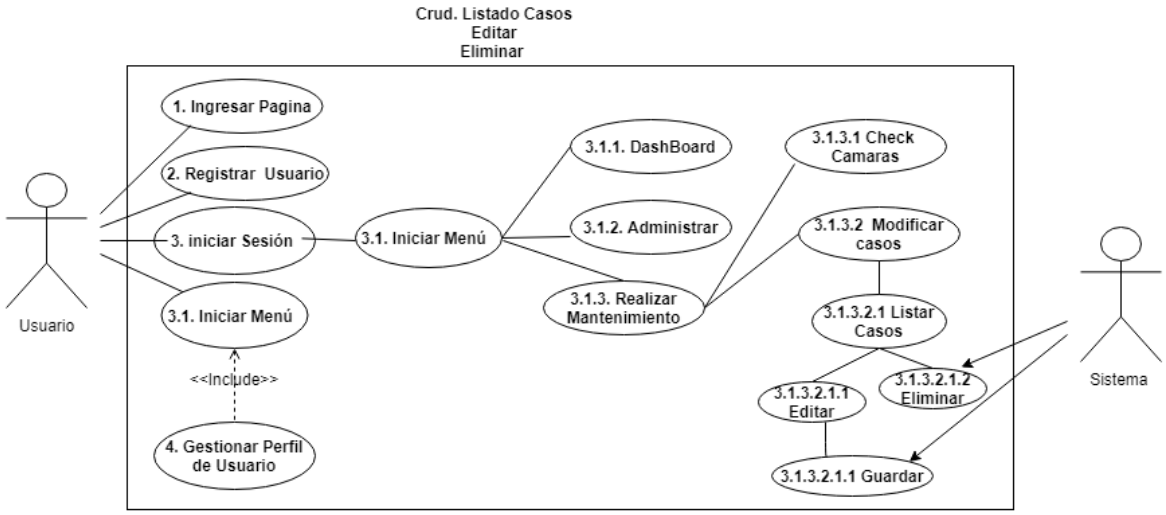


Fig. 13. Diagrama de acceso al sistema de control crud de mantenimiento de eliminar y editar

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso al módulo crud de mantenimiento listas de casos
Actor	Usuarios, Administración
Descripción	Acceder al Sistema y al módulo respectivo
Precondiciones	Ser aceptado por el administrador como root, acceder al módulo de control
Flujo Normal	3.1.3.2 Se ingresa al área de modificar casos 3.1.3.2.1 Listar todos los casos existentes en el sistema 3.1.3.2.1.1 Edita un caso en concreto 3.1.3.2.1.2 Elimina un caso en concreto

Tabla 8. Acceso al módulo de mantenimiento de la lista de casos

En este módulo de control se podrá realizar los diferentes aplicativos del modelo CRUD a la lista de casos que tengamos registrados en el sistema para corregir, eliminar o modificar registros existentes y a su vez crear nuevos, (fig., 13)

Aplicación Móvil

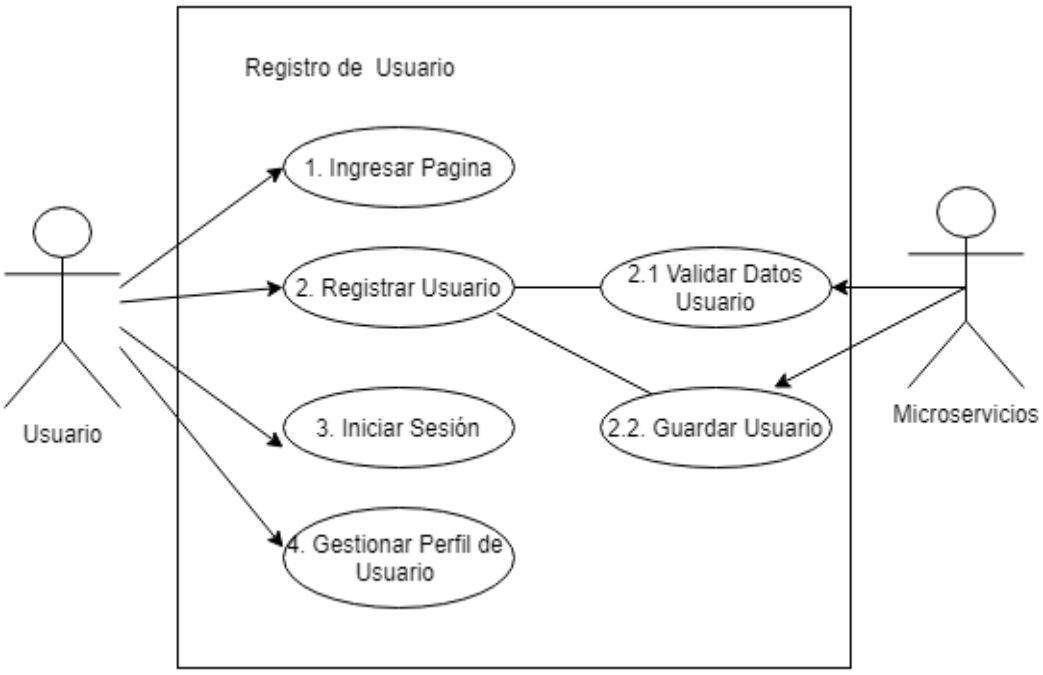


Fig. 14. Diagrama inicio del sistema de aplicación móvil

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Inicio Sistema de aplicación móvil.
Actor	Usuarios, Administración
Descripción	Acceder al Sistema desde la aplicación móvil
Precondiciones	Ser aceptado por el administrador como Root, acceder al módulo de control
Flujo Normal	1. Se ingresa a la pantalla principal de la aplicación móvil 2. Formulario para el registro de usuario 2.1 Ubicación de los datos del usuario antes de guardarlo 2.2 Una vez terminado de realizar la validación se guarda en el sistema 3. Inicio de sesión del usuario 4. Gestionar el tipo de perfil para la sesión del usuario

Tabla 9. Inicio al sistema de la app móvil

Se procede a iniciar la aplicación y visualizar los diferentes entornos de navegabilidad que posee como su área de registro, inicio de sesión y principal que corresponden a su

perspectiva de inicio. (fig., 14)

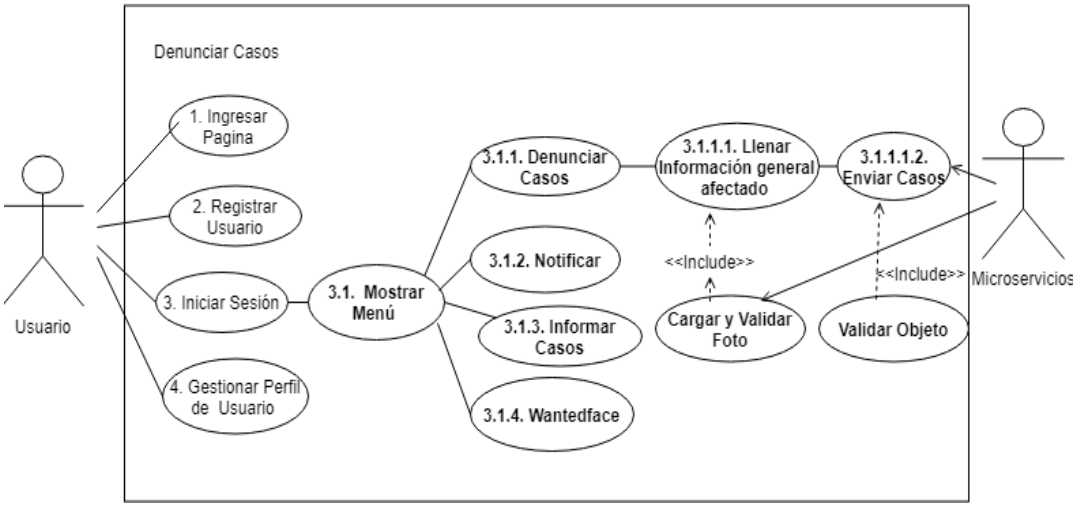


Fig. 15. diagrama de sistema de denuncia casos

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso al Sistema de denuncias de casos
Actor	Usuarios, Administración.
Descripción	Acceder al sistema de denuncias de casos
Precondiciones	Ser aceptado por el administrador como usuario existente, acceder al módulo de denuncias
Flujo Normal	3.1.1 Muestra pantalla principal de la aplicación que es la denuncia 3.1.1.1 Llevar información de un nuevo caso 3.1.1.1.2 La foto debe contener un rostro 3.1.1.2.1 Se valida el caso y se envía al servicio se guarda

Tabla 10. Acceso al sistema de denuncias de casos

En esta sección se procede a registrar casos que el usuario requiera ya sea tomando una foto o escogiendo una de su galería para completar la información requerida al momento de crear un caso para su monitoreo por parte de la aplicación y el sistema inteligente, (fig., 15).

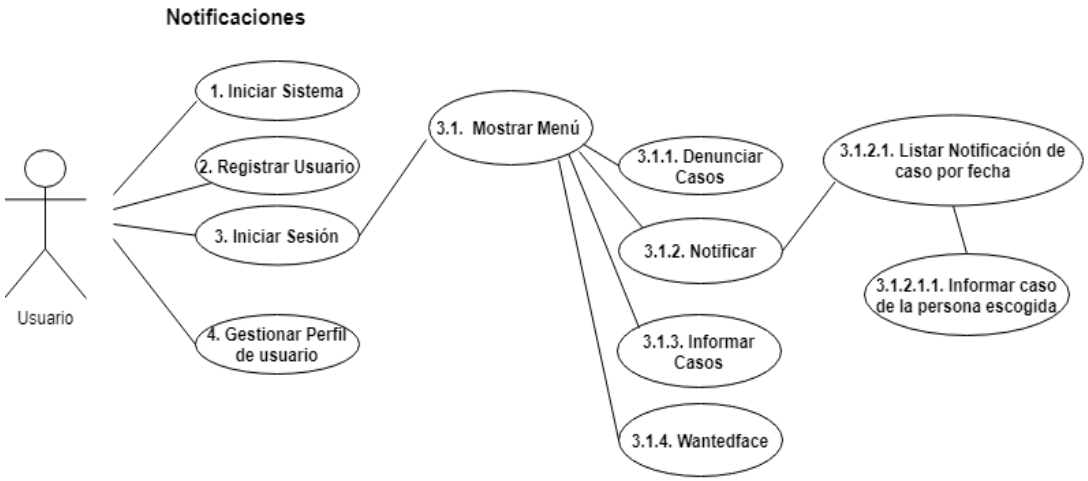


Fig. 16. Diagrama de registro de notificaciones

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso al Sistema de notificaciones.
Actor	Usuarios, Administración.
Descripción	Acceder al módulo del Sistema de notificaciones.
Precondiciones	Ser aceptado por el administrador como usuario existente, acceder al módulo desde el menú
Flujo Normal	1. Se ingresa al menú de la aplicación 2. Se busca la sección de notificaciones 3. Se puede visualizar los registros existentes 4. se selecciona el registro requerido 5. Se accede al mismo y se visualiza su información

Tabla 11. Acceso al sistema de notificaciones

En este aplicativo de la (fig., 16) se puede visualizar el registro de notificaciones que recibiremos en la aplicación móvil de los casos que hemos registrado en el sistema.

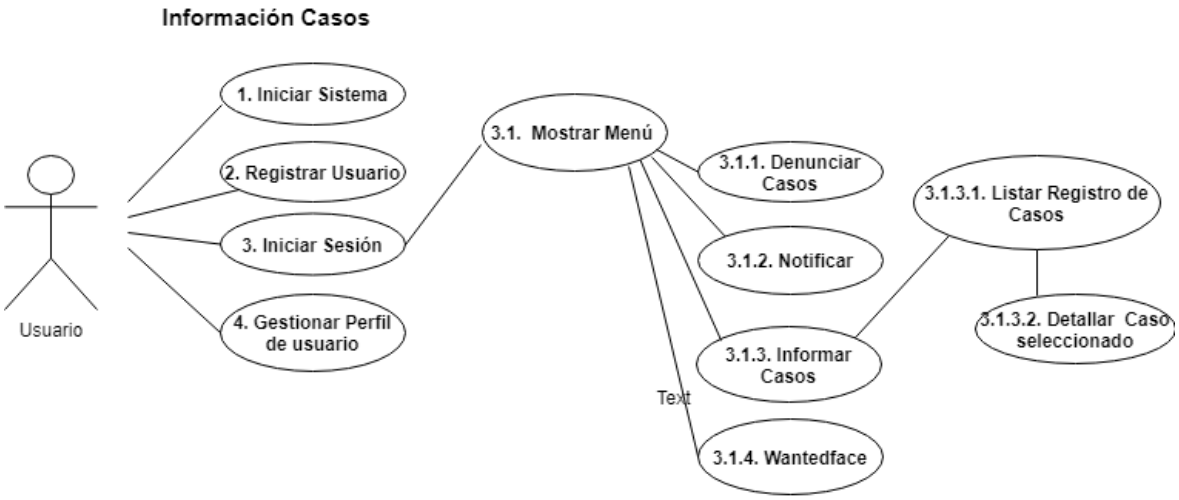


Fig. 17. Diagrama de registro de información de casos

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso al módulo del sistema de información de casos
Actor	Usuarios, Administración.
Descripción	Acceder al módulo informativo de los casos
Precondiciones	Ser aceptado por el administrador como un usuario registrado, acceder al módulo de información
Flujo Normal	1. Se accede a la sección de menú 2. Se ingresa al módulo de información de casos 3. Se puede visualizar los casos registrados 4. Se selecciona el caso buscado y se visualiza su información

Tabla 12. Acceso al módulo del sistema de información de casos

En esta parte de la aplicación móvil, (fig., 17) se procede a consultar en el registro de casos la información correspondiente a cada una de ellas según la seleccionada del listado de casos que se puede tener por usuario en el sistema para lo cual el sistema proporciona la información correspondiente al mismo.

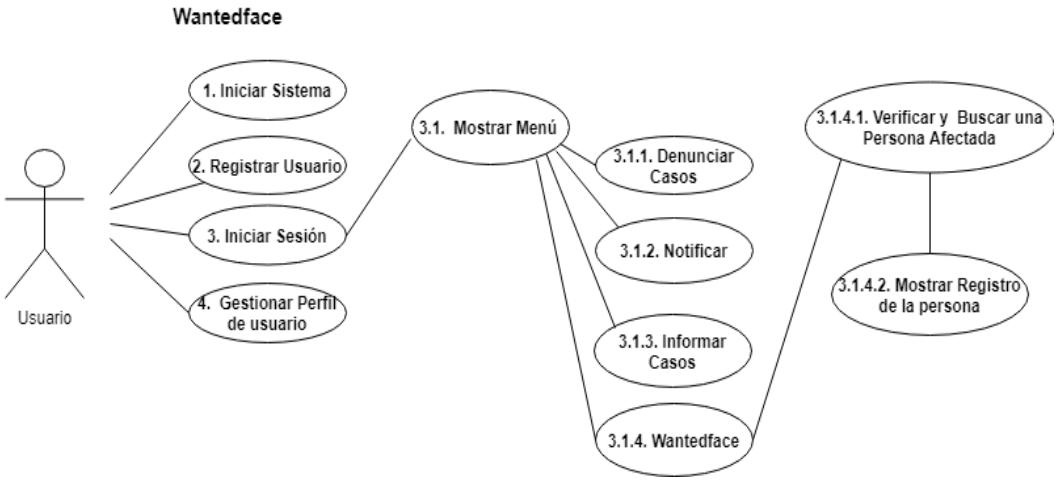


Fig. 18. Diagrama del módulo de wantedface

Fuente: Elaborado por los Autores

Tipo	Descripción
Nombre	Acceso al módulo Wantedface
Actor	Usuarios, Administración.
Descripción	Acceder al módulo del Sistema.
Precondiciones	Ser aceptado por el administrador como usuario registrado, ingresar al módulo
Flujo Normal	1. Se ingresa al menú de navegabilidad de la aplicación 2. Se procede a seleccionar el módulo Wantedface y se ingresa al mismo 3. Se verifica o busca a una persona afectada

Tabla 13. Acceso al módulo wantedface

En esta sección se podrá buscar a una persona afectada ya sea por foto o por registro en el sistema para poder ver su registro existente en el sistema de una manera más precisa como es el reconocimiento facial tal como se indica en la (fig.,18).

7.2 Diagrama de base de datos

Para el diseño o modelo de la base de datos optamos por PostgreSQL, por ser de código abierto ya que usa multiprocesos para atestiguar la escalabilidad del sistema, lo cual será encargado de almacenar todos los datos que son usados dentro del sistema

de reconocimiento facial de los datos del aplicativo móvil y web. Se detalla a continuación el esquema entidad relación para funcionamiento del sistema (fig.,19).

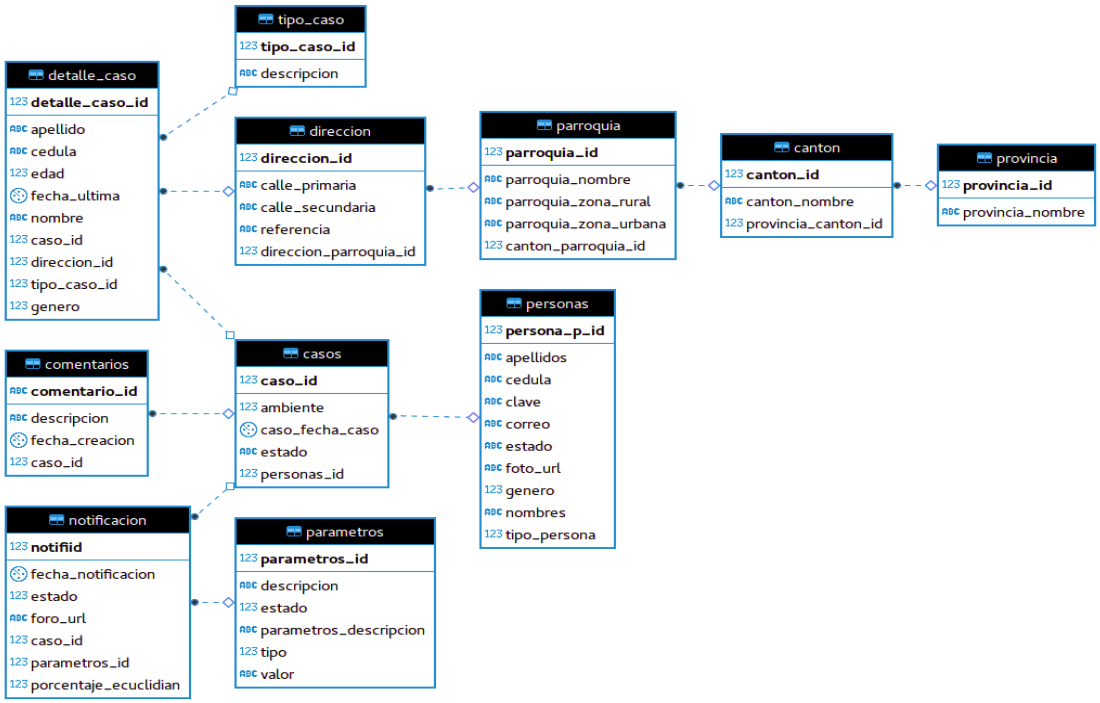


Fig. 19. Diagrama relacional del sistema de la base de datos.

Fuente: Elaborado por los Autores

8. ARQUITECTURA

8.1 Topología y solución a la propuesta

El desarrollo del presente proyecto técnico de reconocimiento facial, fue enfocado para la ciudadanía cuencana en los medios de transportes públicos como una prevención ante los criminales más buscados, secuestradores y en la búsqueda de personas desaparecidas; viendo de una manera factible para la ayuda a los ciudadanos, en vista a esto se desarrolló un aplicativo móvil donde los usuarios podrán interactuar con los siguientes módulos que son: denuncia de tipo de casos, notificaciones, información de casos, también se aplicó una aplicación web donde tiene tres funcionalidades: mantenimiento general, administración cámaras, a través de la cual se pueden gestionar las diferentes áreas, que se encuentra dentro del

sistema como la sala de cámaras, los cuales están bajo la infraestructura de Fog Computing, Cloud Computing, microservicios, inteligencia Artificial. En la presente (fig.,20) se verá el esquema de funcionamiento.

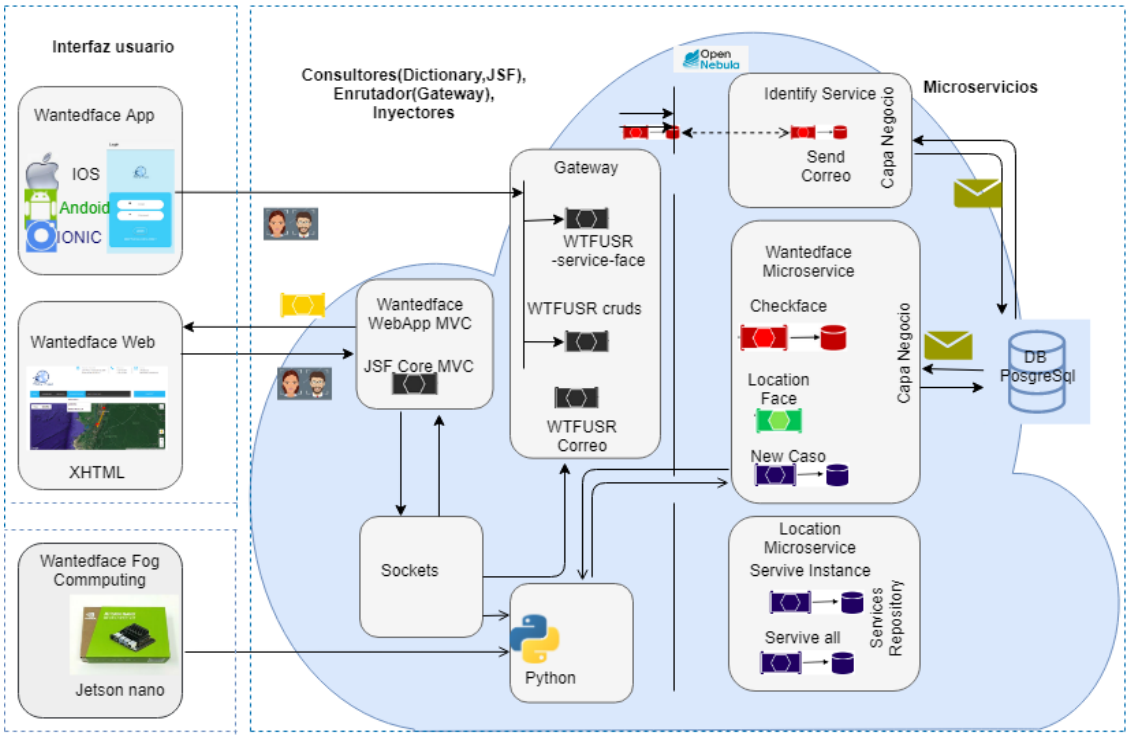


Fig. 20. Esquema de funcionamiento.

Fuente: Elaborado por los Autores

Para el desarrollo del proyecto técnico están involucrados los siguientes elementos en la presente topología.

8.2 Entorno de nube privada: OpenNebula

OpenNebula, es una plataforma de código abierto bajo la infraestructura de IaaS que permite implementar, supervisar y controlar máquinas virtuales que están distribuidas en grupos de recursos físicos. Tiene una capa de abstracción independiente para los servicios subyacentes de redes, virtualización, seguridad y almacenamiento [32]. OpenNebula gestiona y garantiza una variedad de herramientas para creación de una nube propia, esta puede ser privada o hibrida [33].

- ✓ KVM (máquina virtual basada en el kernel): es el hipervisor para la arquitectura de Open Cloud de OpenNebula. Es un sistema de virtualización

para Linux el cual ofrece virtualización completa donde cada máquina virtual interactúa con su propio hardware vitalizado [35].

- ✓ Front: para OpenNebula será una MV (máquina virtual) que migra entre los dos nodos de computo en una disposición de alta disponibilidad. Del mismo modo con el enrutador central y el enrutador de borde [36].

8.2.1 Características de OpenNebula

OpenNebula está enfocado a la virtualización de centro de datos, para construir y gestionar servicios comerciales; para que estos sean flexibles a la innovación de código abierto mediante la escalabilidad. Se observará las características más destacadas [34]. A continuación, se especifica las siguientes características de OpenNebula [34].

- ✓ Monitoreo de los recursos.
- ✓ Gestión de seguridad como pares de claves RSA, ssh, certificados X509 y LDAP.
- ✓ Control avanzado y monitoreo de Infraestructura Virtual.
- ✓ Gestionar host como crear, borrar, activar, desactivar.
- ✓ Soporte de hipervisores Xen, QUEMU-KVM y VMware.
- ✓ Almacenamiento de datos lvm, vmdk.
- ✓ Gestiona los recursos de usuarios, imágenes de máquinas virtuales, plantillas de VM, instancias de VM, redes virtuales, zonas, hosts físicos, autenticación, autorización.

8.2.2 Implementación OpenNebula

Para la implementación de OpenNebula, se usó dos equipos para el funcionamiento de Front y nodo (KVM). En el front se ejecuta los servicios de clúster, y por el lado del nodo habilita el hipervisor para proveer los recursos necesarios para las máquinas virtuales.

Front: será el servidor de openNebula en el cual usará la IP: 192.168.1.10

Nodo(KVM): sobre ello se creará y ejecutará las máquinas virtuales y su dirección de IP: 192.168.1.9

Se detalla a continuación en la tabla de especificación de los requerimientos.

Requerimientos	Configuración usada.
Servidor de Open Nébula (Front-end): Este Servidor permite la administrar y jecutar los servicios de Open Nébula:	• **Procesador:** Intel Core i7 • **Memoria RAM:** 12GB • **Disco Duro:** 240GB • **Sistema Operativo:** Ubuntu 16.04 LTS
Servidor de Open Nébula (KVM No de): Ssistema que se implementó los hipervisores que proporcionan los recursos que necesitan las máquinas virtuales o VPS.	• **Procesador:** Intel Core i7 • **Memoria RAM:** 32GB • **Disco Duro:** 4TB • **Sistema Operativo:** Ubuntu 18.04 LTS Server

Tabla 14. Requerimientos para OPenNebula

8.2.3 Proceso para la instalación de OpenNebula

En esta sección se presentará algunas especificaciones para la instalación de software de OpenNebula, se podrá ver de manera más detallada en siguiente anexo 2 (Manual OpenNebula).

- ✓ Agregar repositorios de OpenNebula.
- ✓ Instalar y configurar la base de datos.
- ✓ Instalar los paquetes Fron-end.
- ✓ Configurar las llaves de acceso de SHH (Secure Shell)
- ✓ Configurar la Red.
- ✓ Configurar el Almacenamiento.
- ✓ Adicionar Host de OpenNebula
- ✓ Verificar la instalación desplegado una máquina virtual

8.2.4 Inicio de sesión de OpenNebula

Ingrese desde un navegador web para el entorno de OpenNebula con la siguiente dirección: htpp://IP-front:puerto/ el puerto 9869 viene pre configurado para seguridad y admiración se debe de cambiar de puerto.

Fig. 21. Inicio de sesión de OpenNebula

Fuente: Elaborado por los Autores

8.2.5 Máquinas virtuales creadas en OpenNebula

El front es la interfaz de OpenNebula (GUI: diseñado para los usuarios y administradores que permitirá la administración de la infraestructura en la nube mediante un explorar web) donde administrara las respectivas máquinas virtuales que se implementó con sus respectivos nombres, las dos máquinas virtuales creadas con su respectivos nombres y sistema operativo GNU/Linux versión Ubuntu 16.04 Server.

- ✓ **VPS1(nombre de la primera máquina virtual que se creó):** se encuentra el gestor de procesos del sistema:
- ✓ **VPS2(nombre de la segunda máquina virtual que se creó)**: se encuentra instalado los requerimientos para Docker para el despliegue del mismo. En presente (fig.,22) se podrá ver la especificación de dos máquinas virtuales.

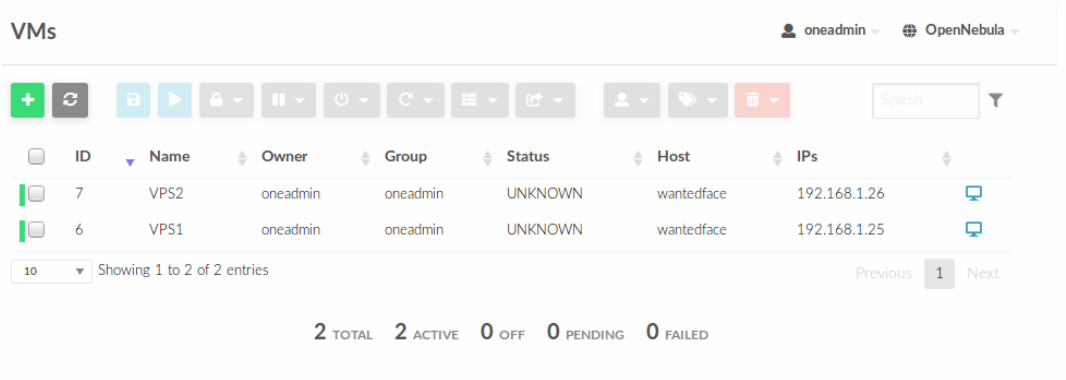


Fig. 22. Máquinas virtuales creadas

Fuente: Elaborado por los Autores

8.3 Interfaz de usuario

Este medio donde el usuario se comunica con la maquina (puede ser, equipo computador) una persona puede controlar un hardware o software especificó.

8.3.1 Implementación de la aplicación web: JSF

Para el desarrollo de la página web optamos por el modelo de n-capas que permite estructurar de una manera ordenada todo el proceso de las fases de desarrollo del proyecto técnico.

8.3.1.1 Java Server Faces (JSF)

El Framework de java es útil para desarrollar aplicaciones web donde simplifica el desarrollo de las aplicaciones a lado del cliente, JSF está basado en la tecnología JEE.
JSF usa Java Server Pages (JPS) tecnología que permite hacer despliegues de las pagina, son adaptables a otras tecnologías como XUL [35].
JSF incluye los siguientes componentes [35]:

- ✓ Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema

de navegación de las páginas y dar soporte para internacionalización y accesibilidad.

- ✓ Un conjunto por defecto de componentes para la interfaz de usuario.
- ✓ Dos librerías de etiquetas personalizadas para JavaServer Pages que permiten expresar una interfaz JavaServer Faces dentro de una página JSP.
- ✓ Un modelo de eventos en el lado del servidor.
- ✓ Administración de estados.
- ✓ Beans administrados.

El presente desarrollo para el Core transaccional (sistema principal) esta implementado con JSF, se verá en la (fig.,23). En el Core transaccional JSF reside toda la lógica de negocios, administra perfil de usuario, ante una sesión de controladores y entidades, este Core también recibe peticiones recurrentes como el referido al mantenimiento de cámaras, aquí se tiene como primer punto clave, ya que no ofrece alta disponibilidad y tolerancia a fallas, tampoco podrá resolver todas las peticiones desde un solo modulo; en este caso, el Core JSF afectaría al rendimiento y colapsaría el sistema, por esa razón segregamos procesos clave en pequeños servicios, que pueden estar disponibles en cualquier momento y en distintas ubicaciones, y, sobre todo, trabajando de manera independiente; cabe recalcar que si se cae uno de los pequeños servicios, los demás siguen disponibles para trabajar.

Estructura del Core transaccional (sistema principal):

- ✓ SGBD
- ✓ Servidor Aplicativo
- ✓ Cliente

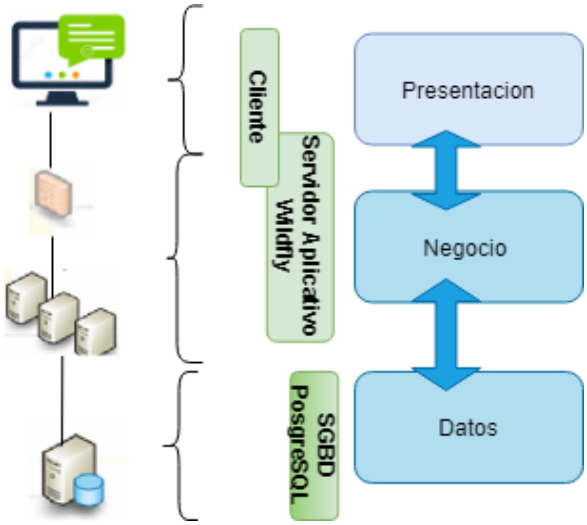


Fig. 23. Arquitectura del core transaccional

Fuente: Elaborado por los Autores

8.3.1.2 Capa de SGB (Datos)

Modelo

Los modelos son objetos que representan a las tablas con su nombre y estos contienen atributos. El modelo JSF está definido mediante beans es similar a lo que utiliza JSP, el beans es una clase que conlleva un conjunto de atributos y métodos de getters y setters (este actualiza y devuelve los valores). se muestra en la (fig.,24) el modelo para registro de usuario.

```java
@Entity
@Table(name="personas")
@NamedQuery(name="Persona.findAll", query="SELECT p FROM Persona p")
public class Persona implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name="persona_p_id")
    private Integer personaPId;

    private String cedula;
    private String nombres;
    private String apellidos;
    private String clave;
    private String correo;
    private String estado;

    @Column(name="foto_url")
    private String fotoUrl;

    @Column(name="tipo_persona")
    private BigDecimal tipoPersona;

    //bi-directional many-to-one association to Caso
    @OneToMany(mappedBy="persona", fetch=FetchType.EAGER)
    @JsonIgnore
    private List<Caso> casos;
    private int genero;
    public Persona() {
```

Fig. 24. Modelo del registro del usuario.

Fuente: Elaborado por los Autores

8.3.1.3 Capa de negocio

En esta capa presenta la lógica necesaria a la capa de presentación, el usuario interactúa con las funcionalidades de la aplicación. En la capa de Servidor Aplicativo o la lógica de negocio es donde se cumple las condiciones recibidas del usuario y envía respuesta donde esta hace una comunicación con la capa de presentación o cliente esta recibe la solicitud y presenta los resultados de la capa de datos donde almacena y recupera los datos del mismo. A continuación, se muestra en la (fig.,25) capa de negocio para registro de usuario.

```java
@Stateless
public class PersonaDAO implements Serializable {

    @Inject
    private EntityManager em;

    public Persona read(Integer personaId){

        Persona aux = em.find(Persona.class, personaId);

        return aux;
    }
    public void save(Persona persona) { em.persist(persona); }

    public void update(Persona persona) {
        em.merge(persona);

    }

    //METODOS CREADOS
    public void remove(Integer personasId){

    }
    public List<Persona> getListaPersona(){
        String jpql = "SELECT distinct p FROM Persona p";
        Query query = em.createQuery(jpql, Persona.class);
        List<Persona> personas = query.getResultList();
        return personas;
    }
```

Fig. 25. Capa de negocio del registro del usuario.

Fuente: Elaborado por los Autores

8.3.1.4 Capa de presentación (Cliente)

En esta capa es la perspectiva del usuario donde se presenta los datos y la navegación de las páginas que permita la entrada de los usuarios a la capa de lógica de negocios.

La capa de presentación donde valida la entrada del usuario y mantiene el estado de sesión de la aplicación.

Las vistas en JSF construye etiquetas específicas como componentes, elementos y utilizan ficheros XHTML. En presente (fig.,26) registro de usuario

```
<h:head>
<link rel="stylesheet" type="text/css" href="resources/css/style3.css" media="screen" />
</h:head>
<body>

<h:form rendered="#{loginController.sesion eq true}">
        <f:metadata>
            <f:viewParam name="paramName" value="#{loginController.sesion}" />
            <f:viewAction action="#{loginController.redirect}" />
        </f:metadata>
</h:form>
<h:form id="f1"  rendered="#{!loginController.sesion}" styleClass="form">
    <p:growl id="growl" showDetail="true" sticky="true" />
    <div class="form">

              <ul class="tab-group">
                <li class="tab active"><a href="#signup">Sign Up</a></li>
                <li class="tab"><a href="#login">Log In</a></li>
              </ul>
        <h:form id="fb">
```

Fig. 26. Vista del registro usuario

Fuente: Elaborado por los Autores

8.3.1.5 Web Service

Web service es el intercambio de la información de equipo a equipo sobre una red en cual un computador o programa pueda solicitar y recibir información de otro computador o programa quien envía la información es el servidor y quien lo solicita la información es el cliente.

Los webs services acceden mediante protocolos web usan XML para el intercambio de la información, usan componentes de aplicación distribuida se puede enviar información desde cualquier lugar del internet y es independiente a la plataforma que se usa como también lenguaje de programación, cuando se conecte a un servidor desde el navegador este devuelve la página solicitada para la visualización para el usuario.

Para la implementación de este proyecto técnico se ha usado webserivice Rest, el cual transporta su información por medio del protocolo de HTTP, posee métodos como seria GET, POST, etc.; y estos se comunican mediante HTTP, transmiten datos de cualquier tipo, permitiendo enviar, XML, JSON, Binarios (Documentos, imágenes), etc. Des de un punto de vista el Rest, que funciona principalmente con el modelo vista controlador(MVC) del lado del cliente (que los procesos se realizarían desde el

navegador).

En la presente (fig.,27), se ve el desarrollo del código web servicie Rest del login de los usuarios para el proceso de logueo.

```
//Webservive for login
@GET
@Path("/login")
@Produces("application/json")
public Persona login(@QueryParam("correo") String correo, @QueryParam("password") String password) {
    Persona per = new Persona();
    String c = new ProcesosReusoCodigo().generaHashMD5(password);
    System.out.println("correo:" + correo);
    System.out.println("password:" + password);
    per = perBus.buscarUser1(correo.trim(), c.trim());
    if (per != null) return per;
      File fi = new File("/home/anita/pink.png");
      try {
          byte[] fileContent = Files.readAllBytes(fi.toPath());
          per.get(0).setFoto(fileContent);
    return null;
      } catch (IOException e) {
          e.printStackTrace();
      }
      return null;
}
```

Fig. 27. Web servicio rest del login usuario

Fuente: Elaborado por los Autores

8.3.1.6 Post

Es parecido al Web rest pero se envía al objeto en el body de la solicitud.

En la (fig.,28) pregunta al servidor si hay un rostro en una imagen esto guarda temporalmente en el path temporal "tempPhotos" para al momento, de tomar la foto desde IOS o Android envía desde un demonio, la foto y chequea si hay un rostro.

```java
@PostMapping("/wtfusr/checkface")
@Consumes(MediaType.APPLICATION_JSON)
public String checkFace(@RequestBody fotoObject object) {
    logger.info(object.toString());
    String url = home + "/tempPhotos/";
    Random rand = new Random();
    int nombreFotoTemp = rand.nextInt(100);
    logger.info(object.getFototemp().split(",")[0]);
    logger.info(object.getFototemp().split(",")[0].split("/")[1]);
    logger.info(object.getFototemp().split(",")[0].split("/")[1].split(";")[0]);

    String extensionFoto = "." + object.getFototemp().split(",")[0].split("/")[1].split(";")[0];
    reuso.decoder(object.getFototemp().split(",")[1], url + nombreFotoTemp + extensionFoto);
    try {
        String urls="http://localhost:5001/temPhoto/" + nombreFotoTemp + "" + extensionFoto;
        logger.info(urls);
        URL uri = new URL(urls);
        HttpURLConnection conn = (HttpURLConnection) uri.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Accept", "application/json");

        if (conn.getResponseCode() != 200) {
            throw new RuntimeException("Failed : HTTP error code : "
                    + conn.getResponseCode());
        }
```

Fig. 28. Webservice post

Fuente: Elaborado por los Autores

8.3.1.7 Get

El método GET hace solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

El servicio web utiliza protocolos HTTP como los métodos GET para obtener datos del servidor web. Cuando el cliente ejecuta una consulta, el servidor responde la consulta con el recurso solicitado por el cliente. Luego puede usar POST, PUT o DELETE para cambiar el objeto

```java
@GET
@Path("/login")
@Produces("application/json")
public Persona login(@QueryParam("correo") String correo, @QueryParam("password") String password) {
    Persona per = new Persona();
    String c = new ProcesosReusoCodigo().generaHashMD5(password);
    System.out.println("correo:" + correo);
    System.out.println("password:" + password);
    per = perBus.buscarUser1(correo.trim(), c.trim());
    if (per != null) return per;

    return null;

}
```

Fig. 29. Webservice método get

Fuente: Elaborado por los Autores

8.3.1.8 Implementación de la página web (JSF)

Dentro de la aplicación web, permite realizar interacción con un usuario administrativo y acceder a cada uno de los componentes que posee la página. Permite de una manera amigable al usuario la intelectualidad para los distintos módulos.

Para el ingreso del sistema, el usuario debe ingresar el correo y la contraseña en la ventana que se presentara. En presente figura (fig.,30) se verá el ingreso al sistema

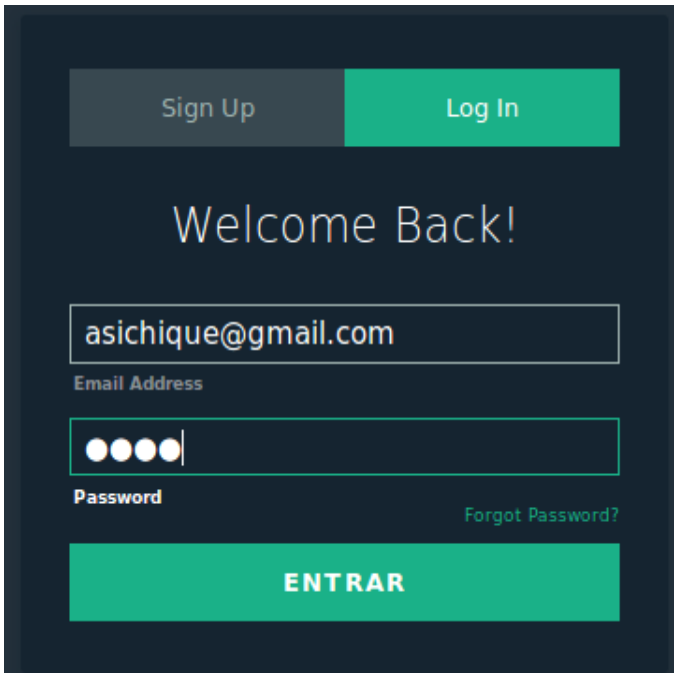


Fig. 30. Ingreso al sistema

Fuente: Elaborado por los Autores

8.3.1.8.1 Administración

Una vez el usuario ingresado al sistema, en el componente administración el administrador de la página web podrá modificar, eliminar casos, verificación de cada una de las cámaras.

En el Menú de Administración tiene 3 submenús que son:

- ✓ Ver Cámaras
- ✓ Ver Fotos

- ✓ Ver Notificaciones

A continuación, se especificará a cada uno de ellos:

8.3.1.8.2 Ver Cámaras

En esta venta se verá el tipo de cámara, a qué red está conectado y su ubicación de las coordenadas geográfica que se encuentra, se puede ver qué cámara está en funcionamiento. En la siguiente figura (fig.,31) se verá cámaras registradas

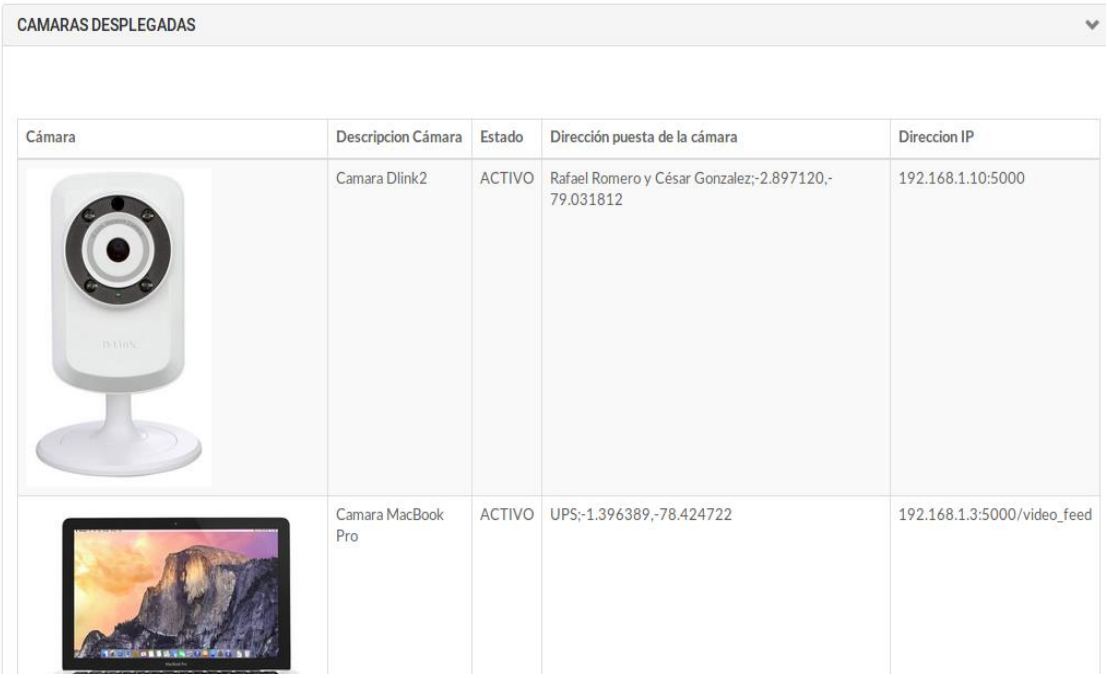


Fig. 31. Cámaras registradas.

Fuente: Elaborado por los Autores

8.3.1.8.3 Ver Fotos

En esta venta se puede ver las fotos de los individuos con sus datos personales que están subidos en la base de datos, para ser procesados y publicados para la búsqueda. En la presente (fig., 32) se mostrará el registro de las fotos del afectado.

Fig. 32. Listados de las fotos del afectado.

Fuente: Elaborado por los Autores

8.3.1.8.9 Ver Notificaciones

En esta ventana se puede ver las notificaciones de los casos, se debe de dirigir en la opción VER, y esto llevará a otra ventana donde se verá las notificaciones más detalladamente de cada uno de los casos. Se mostrará en la (fig., 33) listado de las notificaciones.

Fig. 33. Listado de las notificaciones

Fuente: Elaborado por los Autores

8.3.1.8.9 Proyecto Crud

El componente CRUD(Mantenimiento) tiene dos subcomponentes donde describiremos de cada uno de ello.

- ✓ Mantenimiento Cámara: permite realizar
- ✓ Modificar Casos:

Mantenimiento Cámaras

En la ventana Mantenimiento Cámaras, al terminar de realizar los cambios se da un clikc en la opción ENVIAR los datos se enviará con sus respectivos cambios. En la (fig.,34) se mostrará el mantenimiento de cámaras.

Fig. 34. Mantenimiento de cámaras

Fuente: Elaborado por los Autores

8.3.1.8.10 Modificar Casos

En modificar Casos tiene dos opciones que son:

- ✓ **Editar**: permite hacer alguna modificación del caso de cada componente que tiene cada uno de los casos.
- ✓ **Eliminar**: elimina el caso si desea eliminarlo

En la siguiente (fig., 35) se mostrará el mantenimiento de casos

Fig. 35. Mantenimiento de casos

Fuente: Elaborado por los Autores

8.3.2 Implementación de la aplicación móvil: IONIC

Las aplicaciones móviles estan diseñadas para ser ejecutados en dispositivos móviles como son: tablets, teléfonos.

El desarrollo del aplicativo móvil como cliente bajo el lenguaje de programación IONIC, permite construir software nativo para IOS, Android, Windows Phone. Para la implementación de este proyecto técnico en el aplicativo móvil se realizó el despliegue en la plataforma Android Estudio, IONIC, IOS.

Proceso para la instalación del lenguaje de programación IONIC.

8.3.2.1 IONIC

Es un framework que permite crear aplicaciones hibridas, que al compilar resulta en aplicaciones nativas, tiene una variedad de librerías y herramientas para JS, CSS, HTML, permitiendo que la app se comporte como nativa en el despliegue de cada una de las plataformas, al utilizar versiones como de Android, Google Phone, IOS [47].

IONIC es un lenguaje de programación que permite ejecutar, compilar código en JavaScript a lado del servidor, garantizando que el servidor soporte una variedad de conexiones concurrentes.

8.3.2.2 Comandos usados para Instalación Node JS

#sudo apt-get update

#sudo apt-get install node

#sudo apt-get install nodejs

En la presente (fig., 36) se mostrará la verificación de la instalación nodejs

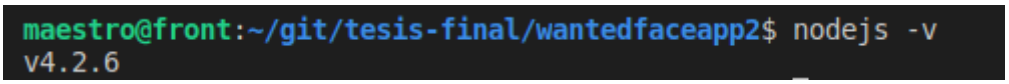


Fig. 36. Verificación de la instalación de nodejs.

Fuente: Elaborado por los Autores

 # sudo npm install –g ionic

Al terminar la instalación de IONIC, se creó la app con el nombre wantedfaceapp2 (nombre que se dio) mediante línea de comando se ve la siguiente línea.

 # inoic start wantedfaceapp2

Para el desarrollo del aplicativo móvil, se realizó las respectivas configuraciones.

Ingrese o diríjase en la siguiente dirección *app.modules.ts* donde se encuentra en la siguiente dirección *src*/app e importar *HttpClienteModule*, permitirá hacer peticiones entre el Core transaccional (sistema principal) con la aplicación, debe hacer la importación y añadir en el aparto imports. En la presente (fig.,37) se mostrará la importación de los componentes.

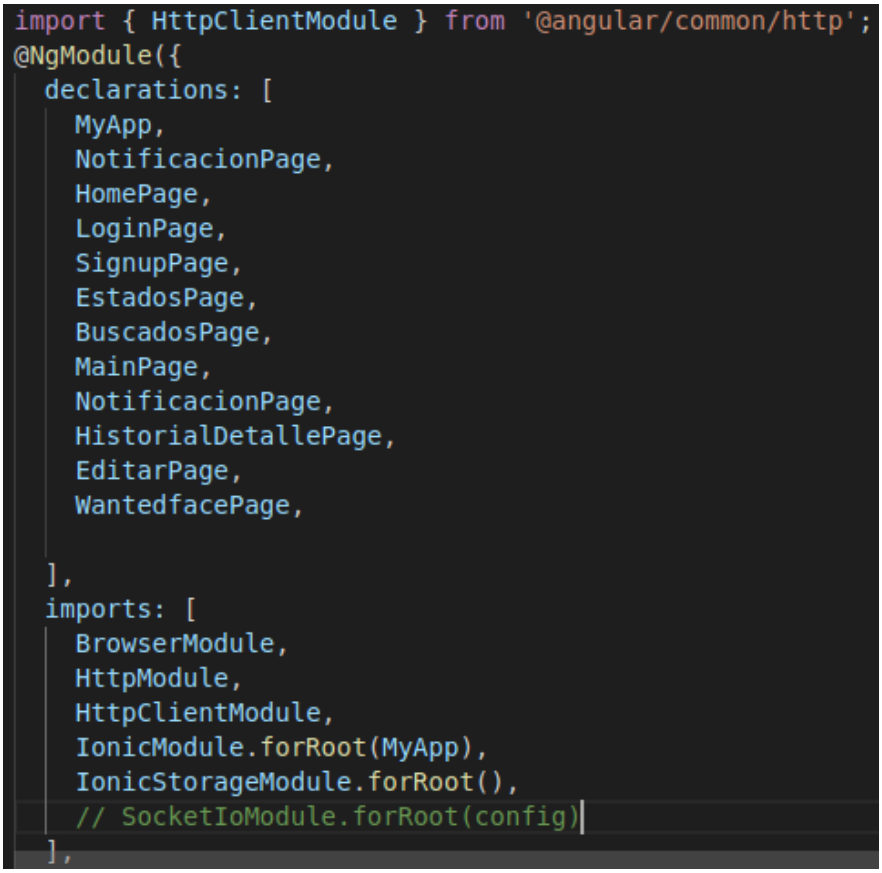


Fig. 37. Importación de los componentes

Fuente: Elaborado por los Autores

Se creara el *service* en la presente dirección src*/providers/core-service* y se importara también el *app.module.ts,* en el *core-service* se creara una variable *apiGet*, se añadirá la *apiGet* de la API y función *getTipoCasos* para hacer la petición, dese el cliente a sia el Core Transaccional(sistema principal). En siguiente (fig.,38) se mostrará peticiones del Core Transaccional.

```
@Injectable()
export class CoreServiceProvider {
  loading: Loading;
  private dispositivo = {
    "saludo" : "hola",
  };
  constructor(public alertController: AlertController,private storage: Storage,public http: HttpClient, pu
    console.log('Hello CoreServiceProvider Provider');
  }
  apiGet:string="http://192.168.1.10:8080/wantedface/wanted/modelos/dota/"
  apiMicro:string="http://192.168.1.10:8082/wtfusr/"
  dologin(email,pass){
    return new Promise(resolve => {
      this.http.get(this.apiGet+"login?correo="+email+"&password="+pass).timeout(10000)
          .subscribe(
           data => {
           resolve(data);
          },
          error => { resolve(error)});

    });
}
```

Fig. 38. Peticiones del core transaccional

Fuente: Elaborado por los Autores

En este método permite traer información desde Core Transaccional (sistema)y saca los datos desde la base de Datos. En la (fig., 39) método para obtener tipo caso.

```
getTiposCasos(){
  return new Promise(resolve => {
    this.http.get(this.apiGet+"listado/tipocasos")
    .subscribe(data => {
          resolve(data);
          });
  });
}
```

Fig. 39. Método para obtención tipo caso

Fuente: Elaborado por los Autores

Mostrará los datos que devuelve el servicio en la page *login* dirija en la siguiente direccion login.ts se siguiente se encuentra *src/pages/login* se importara el servicio en *login.ts*, se creara una variable llamada *registerCredenciales* y un método llamado *getTipoCasos*; donde llamara a la función del servicio enviando *registerCredenciales* y llama al método *getTipoCasos* en la función de IONIC *ionViewLoad*, que se ejecutará la variable en cuanto a nuestra página se cargue. En la (fig.,40) se mostrará obtención de los datos para la vista *login*.

```
@IonicPage()
@Component({
  selector: 'page-login',
  templateUrl: 'login.html',
})
export class LoginPage {
 loading: Loading ;
  registerCredentials = { email: '', password: '' };
  rootPage:any = MainPage;
  constructor(private menu : MenuController,private storage:Storage,private nav: NavController, private auth

  }
  ionViewDidEnter() {
   // the root left menu should be disabled on this page
   this.menu.enable(false);
  }
  ionViewWillLeave() {
   // enable the root left menu when leaving this page
   this.menu.enable(true);
  }
```

Fig. 40. Obtención de los de la vista login

Fuente: Elaborado por los Autores

8.3.2.3 Mostrar datos en la vista

Para mostrar los datos en la vista se debe de haber configurado y desarrollado los pasos anteriores en la variable registerCredenciales se guardará y cargado los datos,diríjase en la dirección de login.html donde se mostrará el título y la información, recomendada por la información del usuario se guarda en la base interna de Android para que no vuelva a loguearse en una segunda ocasión. En la (fig.,41) se verá los datos de la vista login.

```
<ion-row>
  <ion-col>
    <ion-list inset>

      <ion-item>
        <!--Start icon Email-->
        <ion-icon name="mail" item-start></ion-icon>
        <!--End icon Email-->
        <ion-input type="text" placeholder="Email" name="email" [(ngModel)]="registerCredentials.email" r
      </ion-item>

      <ion-item>
        <!--Start icon Eye-->
        <ion-icon name="eye" item-start></ion-icon>
        <!--End icon Eye-->
        <ion-input type="password" placeholder="Password" name="password" [(ngModel)]="registerCredential
      </ion-item>

    </ion-list>
  </ion-col>
</ion-row>

<ion-row>
  <ion-col class="signup-col">
    <!-- <button ion-button class="submit-btn" full type="submit" [disabled]="!registerForm.form.valid">L
    <button ion-button class="submit-btn" (click) ="login()">Login</button>
```

Fig. 41. Muestra los datos de la vista login

Fuente: Elaborado por los Autores

Para la ejecución de la Aplicación una vez realizado todos los pasos anteriores se ingresar en la consola y se ejecutará mediante el comando *# Ionic serve –lab* permitirá abrir la app en el navegador. En la (fig.,42) se mostrará la ejecución de la app móvil.

```
[INFO] Development server running!

       Lab: http://localhost:8200
       Local: http://localhost:8100

       Use Ctrl+C to quit this process

[INFO] Browser window opened to http://localhost:8200!
```

Fig. 42. Ejecución de la app móvil.

Fuente: Elaborado por los Autores

En la presente (fig.,43) se verá el ingreso al aplicativo móvil, el cual el usuario podrá realizar su ingreso a la app.

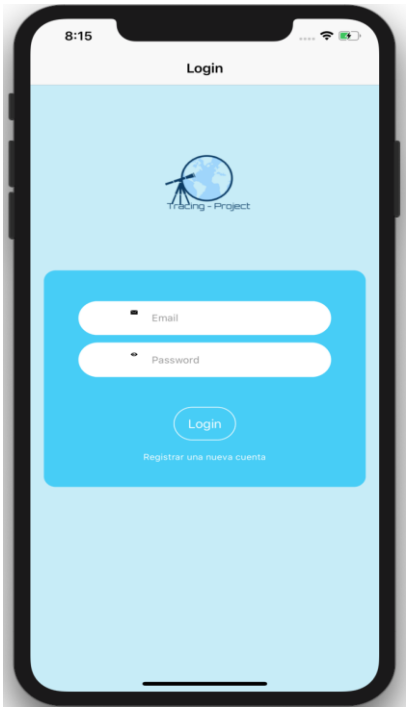


Fig. 43. Ingreso a la app móvil

Fuente: Elaborado por los Autores

8.3.2.4 Configuración en Android

Al termino del proceso y ejecución del aplicativo en el lenguaje de programación en IONIC la funcionalidad correcta.

Se procede la instalación de Android y configuración al tener preparado y listo Android. Ingresamos al terminal de Visual Estudio Code. En la (fig., 44) se mostrará la construcción para Android.

Con el siguiente comando traduce el código IONIC hacia Android nativo.

 # ionic cordova build android

```
maestro@front:~/git/tesis-final/wantedfaceapp2$ ionic cordova build android
> ionic-app-scripts build --target cordova --platform android
[14:55:26]  ionic-app-scripts 3.2.0
[14:55:26]  build dev started ...
[14:55:26]  clean started ...
[14:55:26]  clean finished in 9 ms
[14:55:26]  copy started ...
[14:55:26]  deeplinks started ...
[14:55:27]  deeplinks finished in 173 ms
[14:55:27]  transpile started ...
[14:55:30]  transpile finished in 3.80 s
[14:55:30]  preprocess started ...
[14:55:30]  preprocess finished in less than 1 ms
[14:55:30]  webpack started ...
[14:55:33]  copy finished in 6.42 s
[14:55:35]  webpack finished in 4.17 s
[14:55:35]  sass started ...
[14:55:36]  sass finished in 1.14 s
[14:55:36]  postprocess started ...
[14:55:36]  postprocess finished in 16 ms
[14:55:36]  lint started ...
[14:55:36]  build dev finished in 9.47 s
[14:55:38]  tslint: src/providers/socket-service/socket-service.ts, line: 16
            Property 'storage' is declared but its value is never read.

      L15:  url: string = 'ws://192.168.1.11:8080/wantedface/'
      L16:  constructor(public http: HttpClient,private storage: Storage) {
      L17:    console.log('Hello SocketServiceProvider Provider');

[14:55:38]  tslint: src/providers/socket-service/socket-service.ts, line: 20
            'that' is declared but its value is never read.

      L19:  public conectarServidor() {
      L20:    var that=this
```

Fig. 44. Construcción de la app para Android

Fuente: Elaborado por los Autores

Ingresamos en Android Studio, y como se ve la aplicación desplegada y construida, mandamos a correr. En la siguiente (fig.,45) se mostrará la creación de wantedface en Android.

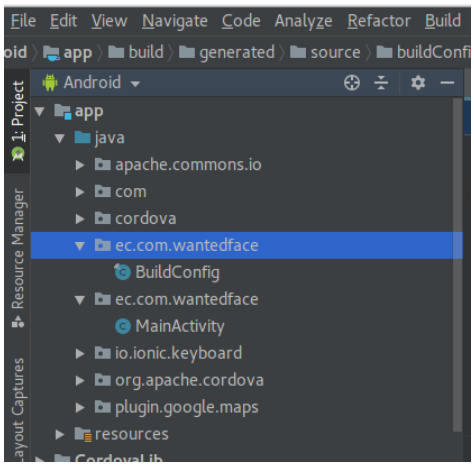


Fig. 45. App wantedface en Android

Fuente: Elaborado por los Autores

Al terminar de

En la presente (fig.,46) se mostrara la ejecucuin de Android se debe de dirigir en la pestaña de color verde y correr, al monto correr se presentara la siguiente otra ventana, para realizar la construcion de la app .

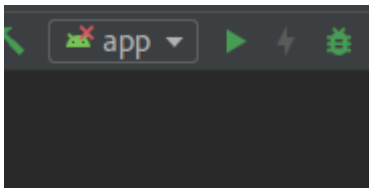


Fig. 46. Ejecución app

Fuente: Elaborado por los Autores

En la (dig.,47) se mostrará la construcción para el Cell Phone de Android, se desplegará la app, una vez creado se podrá ingresar y hacer de su utilidad.

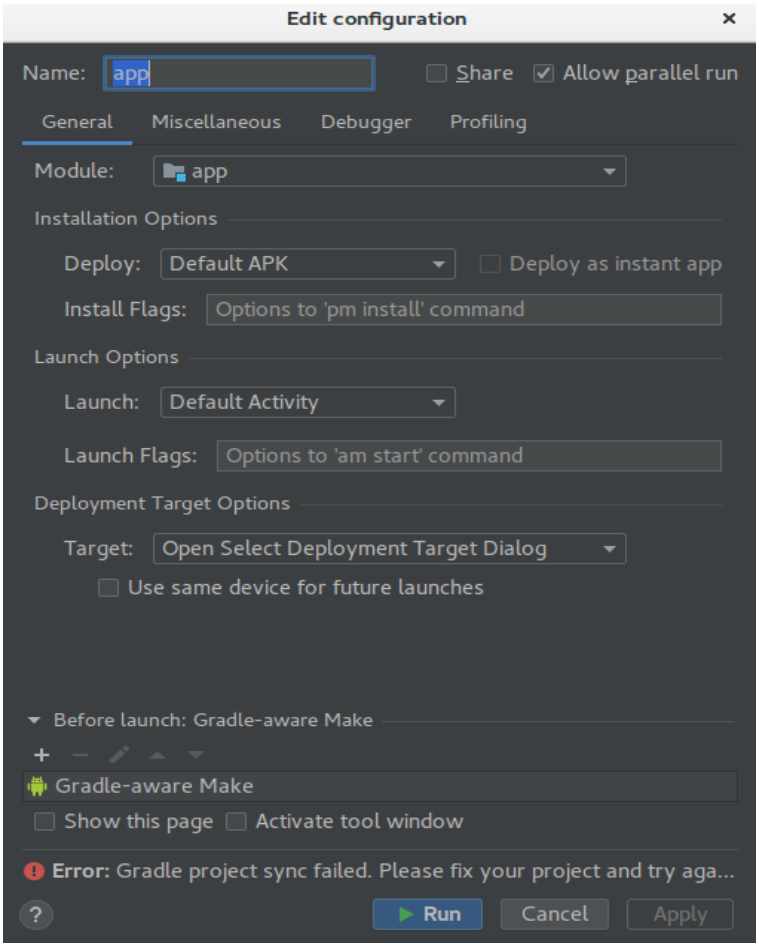


Fig. 47. Construcción de android cell phone

Fuente: Elaborado por los Autores

8.3.2.5 Aplicación Móvil

Como se verá en la (fig.,48) el logo de la app wantedfacce

Fig. 48. Logo app wantedface.

Fuente: Elaborado por los Autores

El aplicativo del presente proyecto, está desarrollado dentro la plataforma de Android, y a continuación se mostrará las funcionales de la aplicación y la usabilidad de cada una ellas.

8.3.2.6 Ventana de registro

En la presente ventana de registro que permite el registro de los usuarios, el usuario debe llenar los datos para el ingreso al sistema. Una vez validado la información procederá el ingreso con los datos registrados anterior mente, este pedirá el correo, contraseña

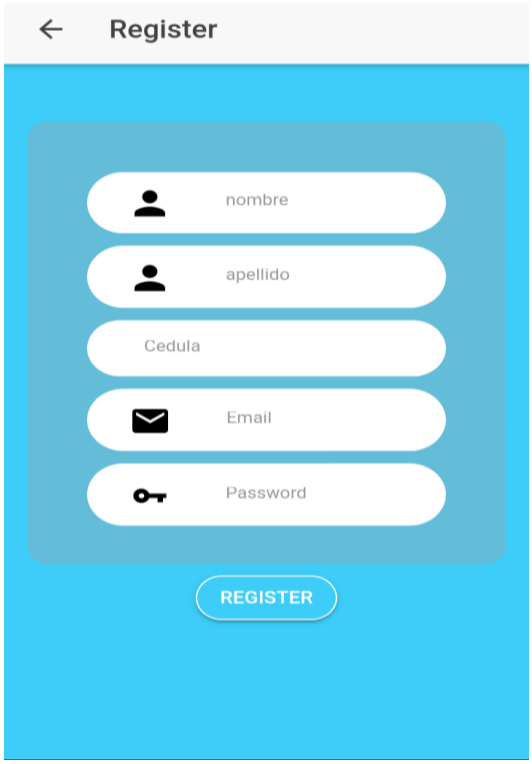


Fig. 49. Registro del usuario app móvil

Fuente: Elaborado por los Autores

8.3.2.7 Menú

Una vez ingresado al sistema se presentará un menú principal que permite la navegabilidad y accesibilidad de cada uno de los aplicativos de la aplicación, como se verá a continuación y se hará una descripción de cada una de las opciones.

- ✓ Denuncia Casos
- ✓ Notificaciones
- ✓ Información Casos
- ✓ Wanted Face

Fuente: Elaborado por los Autores

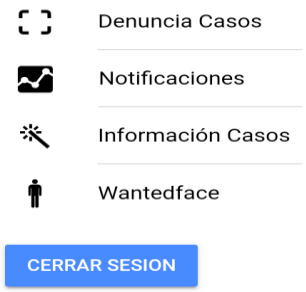


Fig. 50. Menú principal app móvil

8.3.2.8 Denuncia Casos

Permite hacer la denuncia de un caso, el aplicativo por defecto cargará cierta información por defecto del autor del caso como razón social, cedula, fecha y hora del caso, este módulo intercambia información con el Core Transaccional (Sistema Principal) y Microservicios; el flujo de pasos para enviar un caso nuevo es el siguiente:

1. Subir foto: tocamos el ícono de la cámara que mostrará distintas opciones para subir una imagen, desde una foto instantánea o desde una almacenada en el dispositivo, recordando que la imagen seleccionada o tomada debe contener un rostro en caso de tener rostro el dispositivo mostrará una alerta de que la foto se puede usar y se verá en la cabecera del módulo caso contrario se notificará de que debe volver a seleccionar o tomar una nueva imagen.
2. Ubicación del caso y género: se debe ingresar la ubicación para el tipo de caso suscitado como provincia, cantón, parroquia, además de la dirección y

referencias y por último el género de la persona afectada.

3. **Referencias del afectado**: datos o razón social del afectado o el prófugo, edad, cédula del afectado.
4. **Tipo de caso:** escogemos el tipo de caso según la denuncia, éstas pueden ser:
   - ✓ Perdida
   - ✓ Secuestro
   - ✓ Más buscado

Al Terminar de hacer la denuncia enviamos el caso.

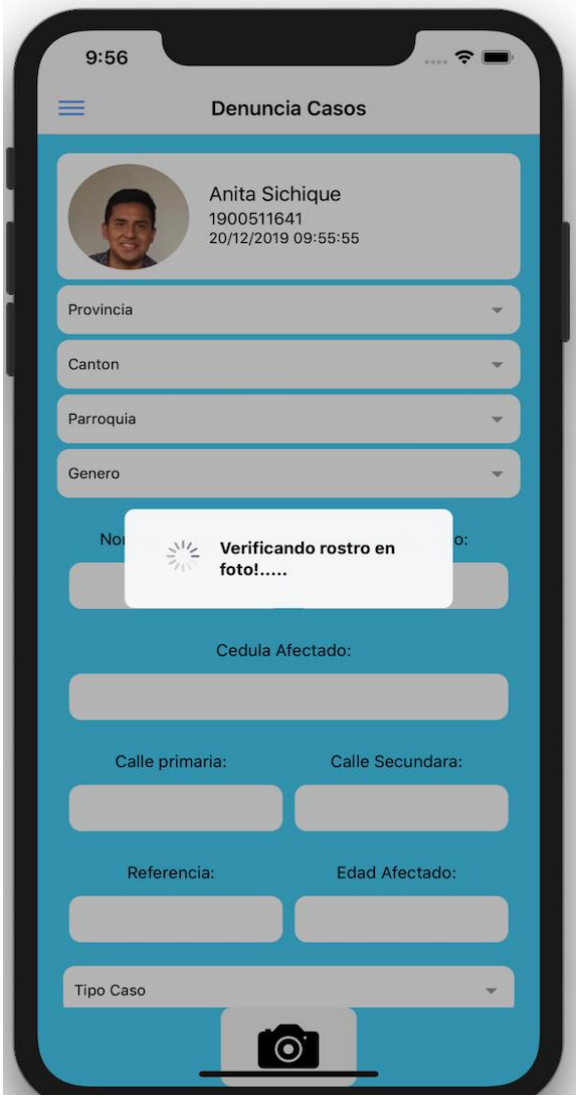


Fig. 51. Denuncia de casos app móvil.

Fuente: Elaborado por los Autores

8.3.2.9 Información Casos

Se refiere a información casos, muestra la foto y los datos personales del individuo, la fecha de la que fue hecha la denuncia y el tipo de caso ya sea una de persona extraviada, más buscada o prófugo.

Donde listara todos los casos de cada uno de ellos detalladamente.

Wantedface

Módulo el cual nos permite tomar una foto de un individuo para verificar el estado de tal rostro o persona, es de utilidad ya que se puede usar independientemente de un caso, esto quiere decir que podemos hacerlo de cualquier persona, el resultado en caso de ser un caso conocido mostrará la razón social y el tipo de caso.

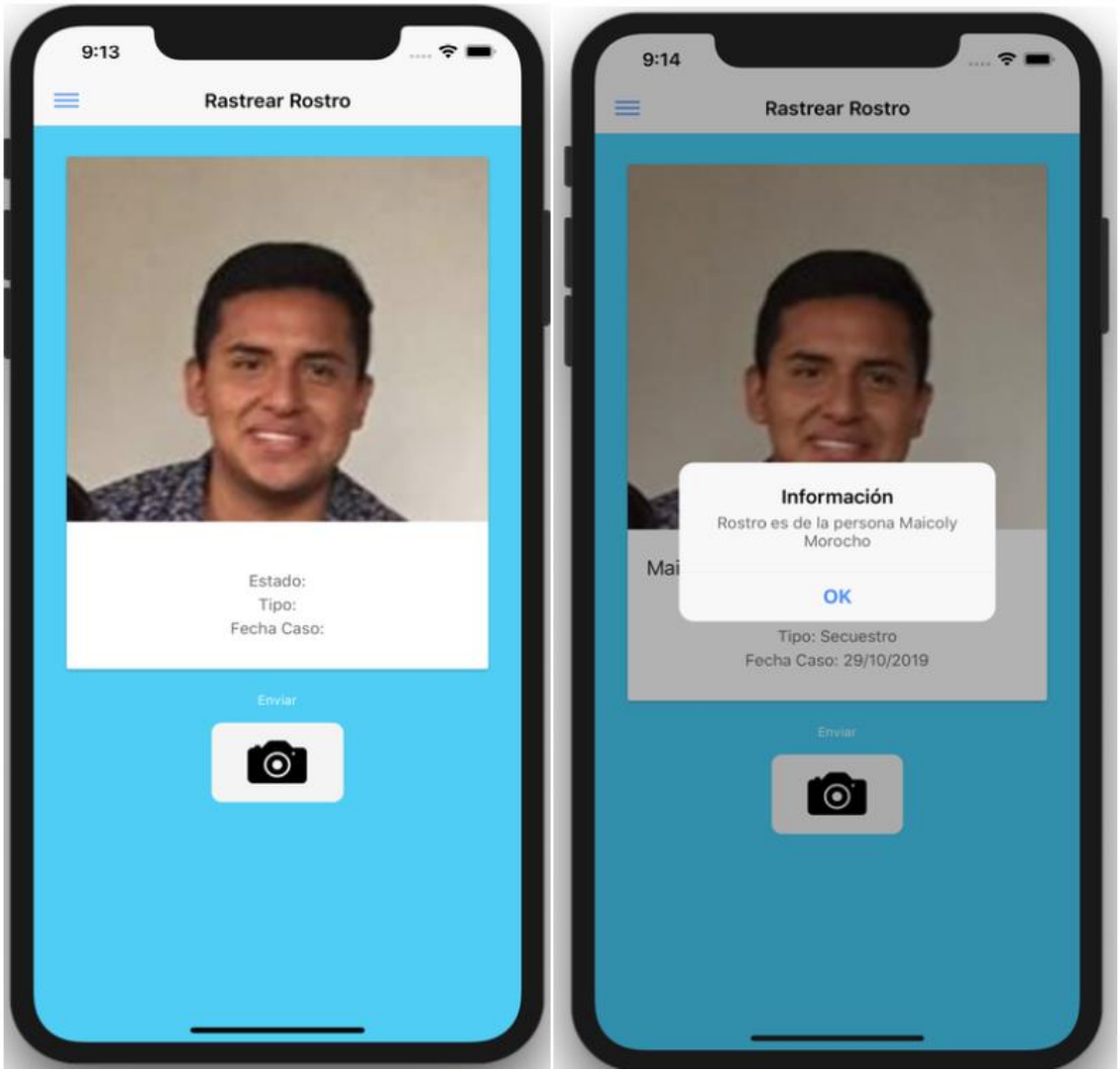


Fig. 52. Rastreo de rostros app móvil.

Fuente: Elaborado por los Autores

8.3.3 Fog Computing

8.3.3.1 Esquema Físico

Fog Computing permite que los datos estén almacenados en los mismos dispositivos inteligentes en lugar de que lo estén en la nube.

Mediante el análisis de los requerimientos de software y hardware, que realiza la interacción del esquema entre los diferentes componentes. En la (fig., 53) se verá los elementos del hardware utilizado en el proyecto y la comunicación con cada uno de ellos.

Jetson Nano de Nvidia es capaz de usar la GPU (Unidad de Procesamiento Gráfico) y la CPU (Unidad Central de Procesamiento) de una manera adecuada a los requerimientos.

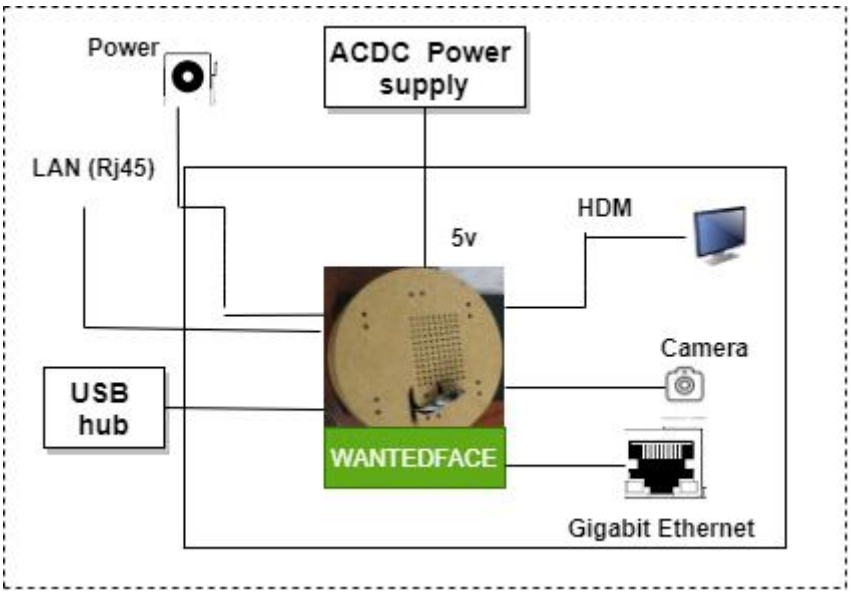


Fig. 53. Esquema físico del proyecto.

Fuente: Elaborado por los Autores

8.3.3 2 Esquema lógico

Esta arquitectura como cliente validara el componente electrónico, el cual tiene una lógica de programación embebida usando inteligencia artificial (IA) y sockets, su

funcionalidad captura cada cambio de frame en la cámara, si logra detectar un rostro conocido empaqueta en el objeto caso donde hace una notificación al servidor principal y a los clientes los que están usando socktes.

El servidor Wildfly y los sockets, estos realizan sus tareas adecuadas haciendo uso de cada uno del servicio requerido de acuerdo a la tarea se requiera la petición.

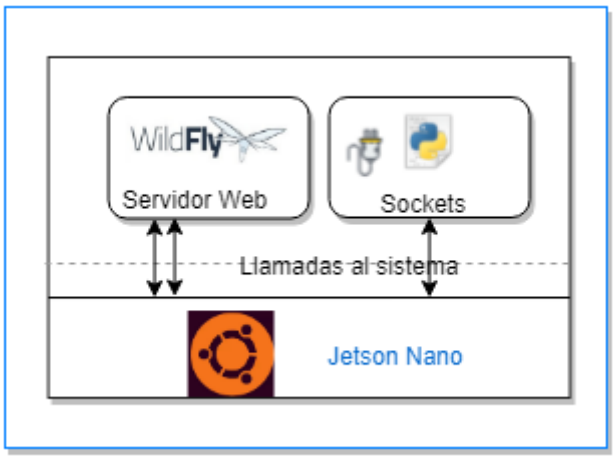


Fig. 54. Esquema lógico del sistema

Fuente: Elaborado por los Autores

8.3.3.3 Cliente Sockets Python

En el servidor importamos los sockets para acceder todas las funcionalidades para el proceso del desarrollo de los mismos, se debe de crear un socket cliente, luego se hará la conexión con el servidor y procederá la comunicación entre los dos.

- ✓ Inicia sesión en el servidor web socket
- ✓ Determina frame y envía el objeto al server web socket

Se habla del objeto frame capturado desde Python y se envía al web socket

8.3.3.4 Como funciona

Los sockets constituyen dos programas uno para el cliente y el otro para el servidor en donde el servidor “escucha” en determinado puerto (se debe especificar), y el cliente debe de conocer la IP “o nombre de dominio/hostname del servidor y el puerto que está escuchando y solicita al establecer una conexión con el servidor”, si el servidor

admite la conexión y habrá conexión entre ellos [43].

8.3.3.5 Código clase principal

En esta (fig.,55) es la clase principal donde las demás clases hace una instancia se ejecutará.

```
from flask import Flask, render_template, Response, jsonify
from camera import VideoCamera

app = Flask(__name__)
aux=5
@app.route('/')
def index():
    return render_template('index.html')

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    update()
    return Response(gen(VideoCamera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/updatefiles')
def update():
        VideoCamera().update()
        return jsonify({'result':'true'})

if __name__ == '__main__':
    app.run(host='0.0.0.0',debug=True,port=5000)
```

Fig. 55. Clase principal Python

Fuente: Elaborado por los Autores

8.3.3.6 Código para cámara

En esta (fig.,56) se encuentra la lógica de programación de la cámara de video, el proceso de cada método en el desarrollo.

```
imagenes=[]
codificado=[]
known_face_encodings=[]
known_face_names=[]
files=[]
r=random.randint(0, 100)
socket=socket.gethostbyname(socket.gethostname())
ws = create_connection("ws://192.168.1.10:8080/wantedface/server/"+str(socket)+",py,"+str(socket))
cont=True
#Coneccion url foto
url='http://192.168.1.11/image/jpeg.cgi'
username = 'admin'
password = '251207'
password_mgr = urllib.request.HTTPPasswordMgrWithDefaultRealm()
password_mgr.add_password(None, url, username, password)
handler = urllib.request.HTTPBasicAuthHandler(password_mgr)
opener = urllib.request.build_opener(handler)
class VideoCamera(object):
    def __init__(self):

        self.video = cv2.VideoCapture(0)
#       video_capture = cv2.VideoCapture(0)

    def sendDataTOWS(self, idDcaso,idPersona, porcentage):
        global r
        global ws
        global socket
        aux=idDcaso+";"+str(idPersona)+",py,"+str(round(porcentage,2))
        ws.send(aux)

    def update(self):
        global files
        global codificado
        global known_face_encodings
        global known_face_names
        global imagenes
        global cont
        global r
```

Fig. 56. Lógica de programación video cámara

Fuente: Elaborado por los Autores

8.3.3.7 Levantamiento de servicios

Permite el levantamiento de los servicios de inteligencia Artificial (IA), estos servicios se encargan de las peticiones de la aplicación móvil.

```
maestro@front:~/git/tesis-final/addons$ python3 web1.py
 * Serving Flask app "web1" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5001/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 985-018-345
```

Fig. 57. Consumiendo servicio

Fuente: Elaborado por los Autores

8.3.3.8 Detección de reconocimiento Facial

Además, tenemos servicios llamados addons que recepta peticiones de los microservicios desde la aplicación móvil como detección de rasgos faciales. Este servicio ayuda a determinar al administrador para que pueda validar un caso siempre y

cuando exista un rostro en la foto enviada y así desplegar a las cámaras.

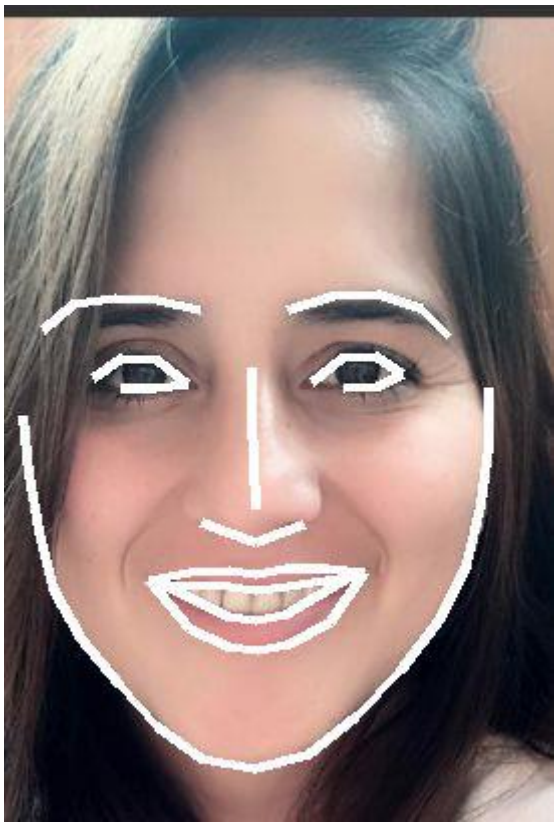


Fig. 58. Detección del rasgo facial.

Fuente: Elaborado por los Autores

8.4 Consultor, Enrutador, Inyector.

8.4.1 Consultores

Para esta interrogante cuenta con el diccionario, más conocidos como Eureka y Gateway Zuul, donde cada petición entrante al router o Gateway es filtrada para este, que solo enruta y no sabe cómo llevar tal petición al servicio correcto; es aquí donde el diccionario o Eureka toma partido comunicando al router la ubicación del servicio que requiere tal petición, una vez que este responde al router, este enviará al receptor y transmitirá a la fuente de la solicitud. Conociendo el flujo de funcionamiento mencionado, se afirma que el router recepta las solicitudes de la aplicación móvil y la aplicación web siempre y cuando sean peticiones a microservicios, ya que pueden existir peticiones como, por ejemplo, de mantenimiento de cámaras; donde se puede

cambiar el nombre o IP de una cámara y no requerir microservicios, sino directo el Core transaccional en JSF. Si deseamis levntar la aplicación en el puerto 9898. Despues indica donde se encuetra Eureka y como deseamos que nos descubra (localhost), En la presente (fig.,59) se muestra la configuración de Eureka.

```
eureka:
  instance:
    hostname: localhost
  client:  # Not a client, don't register with yourself
    registerWithEureka: false
    fetchRegistry: true #poner true para facturacion electronica
    serviceUrl:
      defaultZone: http://mmorocho:251207@localhost:9998/eureka
  renewalPercentThreshold: 0.50
server:
  port: 9998   # HTTP (Tomcat) port
  eviction-interval-timer-in-ms: 1000
# Discovery Server Dashboard uses FreeMarker.  Don't want Thymeleaf
spring:
  thymeleaf:
    enabled: false     # Disable Thymeleaf

logging:
  root: info
#  file: /home/maestro/mikrom/logs/discovery.log
  file: /opt/jetty/logs/micro/eureka.log
  pattern:
    console: "%d %-5level %logger : %msg%n"

management:
  security:
    enabled: false
```

Fig. 59. Configuración EUREKA

Fuente: Elaborado por los Autores

8.4.2 Enrutador (Gateway)

Al usar un API Gateway para los microserviciso permite reducir los esfuerzos de programación permitiendo las aplicaciones sean más eficientes, suministra una interfaz de API REST o WebSocket, este hace enrutador, brindando un punto de entrada a un grupo de microservicios mediante la puerta de enlace “Gateway”, donde cada petición entrante al router o Gateway es filtrado por este, cuando no sabe cómo llevar tal petición a la ubicación del servicio que requiere tratamiento, sino que

pregunta al diciconario, una vez que este responda al router, enviara al receptor y transmitirá a la fuente de la solicitud. El API Gateway separa las API publicas externas de las API internas de los microservicios, de igualmanera oculta el descubrimiento de los servicios publicados en el API principal, donde tiene un único punto de entrada para los microservicios y API's. no solo sirve como puerta de enlace, sí no que también permite agregar funcionalidades adicionales serian seguridad, administración del ciclo de vida y versionado de la API, balanceo de carga, etc.

8.4.3 Inyectores (WebSocket)

Permite comunicar entre el cliente y el servidor a través de la conexión TCP, este hace interacción con una solicitud HTTP, la página web pude ser dinámica e interactiva, su combinación de transmisión HTTP y Ajax con una solución simple y efectiva. El Sring Framerwork tiene un API de websocket, permite realizar aplicaciones de ha lado del servidor y cliente el cual manejan mensajes de websocket [46]. Por ejemplo, en la presente imagen se verá (fig.,60) la configuración de websocket.

El Core transaccional alberga Sockets, este recibe información de un caso desde un frame detectado en la cámara, el cual revisa todas las sesiones activas en el aplicativo web, y envía las notificaciones por sockets; es importante destacar que estas notificaciones se hacen en tiempo real. Al final, el Socket intercambia información indistintamente de donde proviene la información o a donde envía; para el caso que proviene desde Python e inyecta las notificaciones a JavaScript para mostrar en la página web la detección.

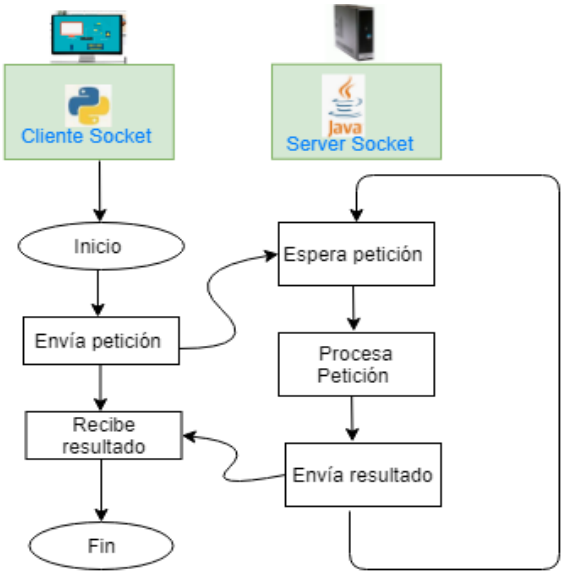


Fig. 60. Peticiones del servidor y cliente socket

Fuente: Elaborado por los Autores

Creación de socket en java, para la conexión y comunicación con clientes de Python. En la (fig.,61) como se verá la codificación web socket server.

```
public class Client {
    final static CountDownLatch messageLatch = new CountDownLatch(1);
    public Session setConnect(String Auth, String server) {
        try {
            WebSocketContainer container = ContainerProvider.getWebSocketContainer();
            String uri = "ws://"+server+":8080/wantedface/server/"+Auth;
            Session session=container.connectToServer(WebSocketClientEndpoint.class, URI.create(uri));
            return session;
//                  session.getBasicRemote().sendText(Auth);
//                  container.connectToServer(WebsocketClientEndpoint.class, URI.create(uri));
//                  messageLatch.await(100, TimeUnit.SECONDS);
        } catch (DeploymentException | IOException ex) {
            Logger.getLogger(Client.class.getName()).log(Level.SEVERE, msg: null, ex);
        }
        return null;
    }
```

Fig. 61. Web socket server

Fuente: Elaborado por los Autores

Web socket contiene métodos definidos para la comunicación entre el servidor con los distintos clientes en ellos están:

Abrir una sesión

Guarda una sesión desde Python, java script

```
@OnMessage
public void onMessage(String message, Session session,@PathParam("key") String key)throws IOException,InterruptedExce
//        System.out.println("Recibido desde onMessage: " + message);
        if(message.length()>2)
        for(Map.Entry<String, Session>entry:sesiones1.entrySet()) {
            Session s=entry.getValue();
//          System.out.println(entry.getKey()+" "+s.getId());
            String[] valores = entry.getKey().split( regex: ",");
            String[] backend = message.split( regex: ";");

            if(backend.length==2 &&( valores[1].equals("f") ||valores[1].equals("mov"))) {
                System.out.println("Se debe envia a:"+entry.getKey());
                if(valores[1].equals("f"))
                sesiones1.get(entry.getKey())
                            .getBasicRemote()
                            .sendText( s: backend[0]+";"+message.split( regex: ",")[2]);
                if(valores[0].equals(backend[1].split( regex: ",")[0])){
                    System.out.println("Se debe envia al movil:"+entry.getKey());
                    sesiones1.get(entry.getKey())
                            .getBasicRemote()
                            .sendText( s: backend[0]+";"+message.split( regex: ",")[2]);
                }
```

Fig. 62. Método abrir sesión

Fuente: Elaborado por los Autores

Enviar sesión

Envía un cliente se desconecta o quita la sesión

```
@OnOpen
public void onOpen(Session session,@PathParam("key") String key) {
//        System.out.println("Conexion abierta "+key+" sesion "+session);
        String username =(key);
        String a[]=key.split( regex: ",");
        if(!a[0].equals("undefined") && !a[0].equals(null) && !a[0].equals(""))
            sesiones1.put(key, session);
        System.out.println ("Lista de sesiones:");
        for(Map.Entry<String, Session>entry:sesiones1.entrySet()) {
            System.out.println("-->"+entry.getKey());
        }
```

Fig. 63. Método que envía sesión
Fuente: Elaborado por los Autores

Enviar mensaje

Envía datos o notificaciones a los clientes

```
@OnMessage
public void processMessage(String message) {
    System.out.println("Mensaje recibido de cliente: " + message);
    Client.messageLatch.countDown();
}
```

Fig. 64. Método que envía notificaciones

Fuente: Elaborado por los Autores

8.5 Implementación de los microservicio

El sistema está implementado con microservicios usando el servidor Eureka bajo el Spring Cloud, esto nos quiere decir que procesos y servicios grandes (problemas) lo podemos dividir en pequeños microservicios independientes desplegados en diferentes nodos y corriendo en el mismo puerto o diferentes.

Antes de analizar los microservicios se debe mencionar ciertos componentes o proyectos necesarios para tener una arquitectura de microsevicios en la nube.

EUREKA Discovery

Eureka es un servidor que se comporta como una guía telefónica donde residen o donde se pueden localizar los microservicios, además de brindarnos información de estos como estado, metadata (XML). Una vez registrado cada instancia de microservicio este emitirá la información a ZUUL (Router) cuando éste lo solicite.

Cada microservicio emitirá un impulso (hearbeat) de su estado cada cierto tiempo configurable, hace 3 intentos en caso de no responder se quitará de EUREKA.

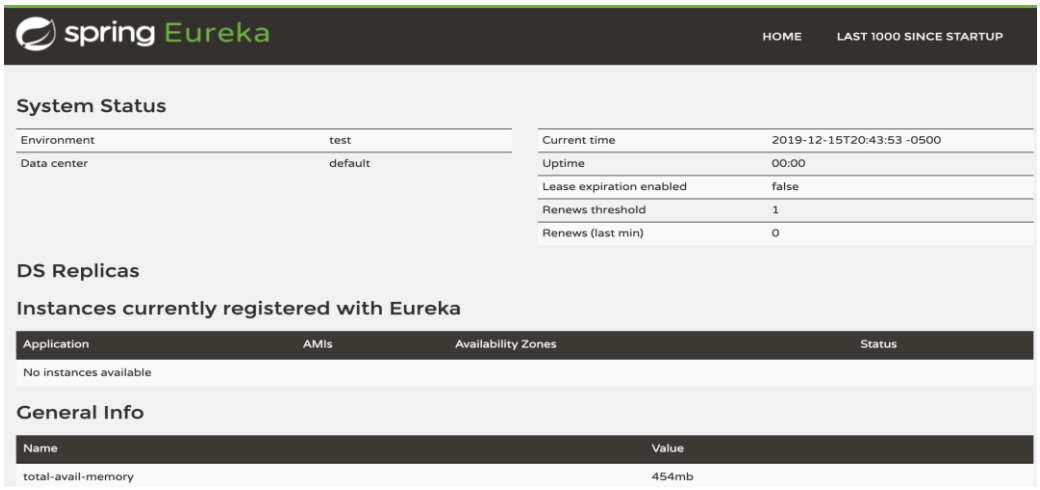
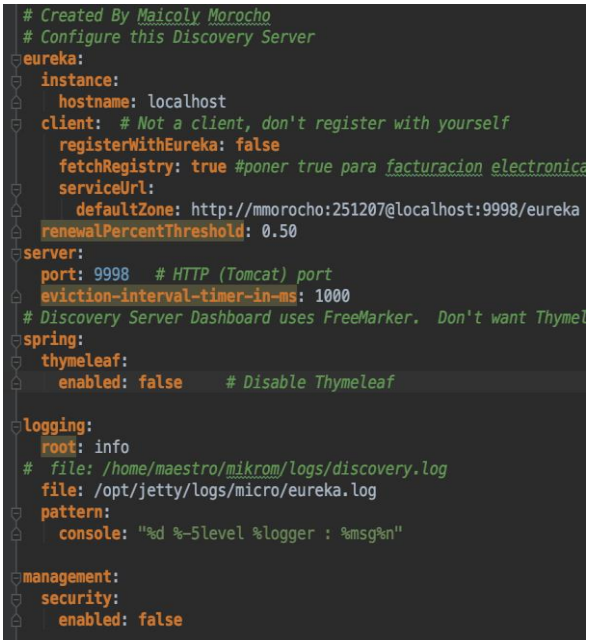


Fig. 65. Distribuidor de servicios EUREKA

Fuente: Elaborado por los Autores

Configuración de EUREKA (application.yml)

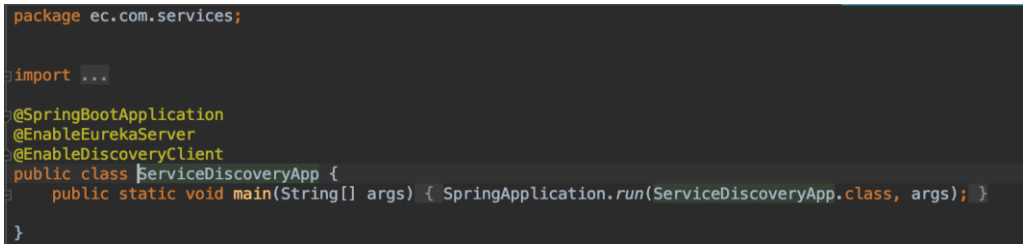


```
# Created By Maicoly Morocho
# Configure this Discovery Server
eureka:
  instance:
    hostname: localhost
  client:  # Not a client, don't register with yourself
    registerWithEureka: false
    fetchRegistry: true #poner true para facturacion electronica
    serviceUrl:
      defaultZone: http://mmorocho:251207@localhost:9998/eureka
  renewalPercentThreshold: 0.50
server:
  port: 9998   # HTTP (Tomcat) port
  eviction-interval-timer-in-ms: 1000
# Discovery Server Dashboard uses FreeMarker.  Don't want Thymel
spring:
  thymeleaf:
    enabled: false     # Disable Thymeleaf

logging:
  root: info
#  file: /home/maestro/mikrom/logs/discovery.log
  file: /opt/jetty/logs/micro/eureka.log
  pattern:
    console: "%d %-5level %logger : %msg%n"

management:
  security:
    enabled: false
```

Fig. 66. Configuración de discovery server.

Fuente: Elaborado por los Autores

ServiceDiscoveryApp.java



```
package ec.com.services;


import ...

@SpringBootApplication
@EnableEurekaServer
@EnableDiscoveryClient
public class ServiceDiscoveryApp {
    public static void main(String[] args) { SpringApplication.run(ServiceDiscoveryApp.class, args); }

}
```

Fig. 67. clase principal del descubridor.

Fuente: Elaborado por los Autores

ZUUL (Router)

Este proyecto se comporta como un Router ya que recibe, en ruta las solicitudes o request y filtra pudiendo aceptar o rechazar de acuerdo a nuestra configuración de seguridad, en conclusión, recibe las peticiones, pregunta a EUREKA la localización de los microservicios y enruta a los servicios que se quiera consumir.

Además, cuenta con un servicio para el balanceo de carga y tolerancia a fallos en caso de que una instancia de microservicio falle llamado Hystrix.

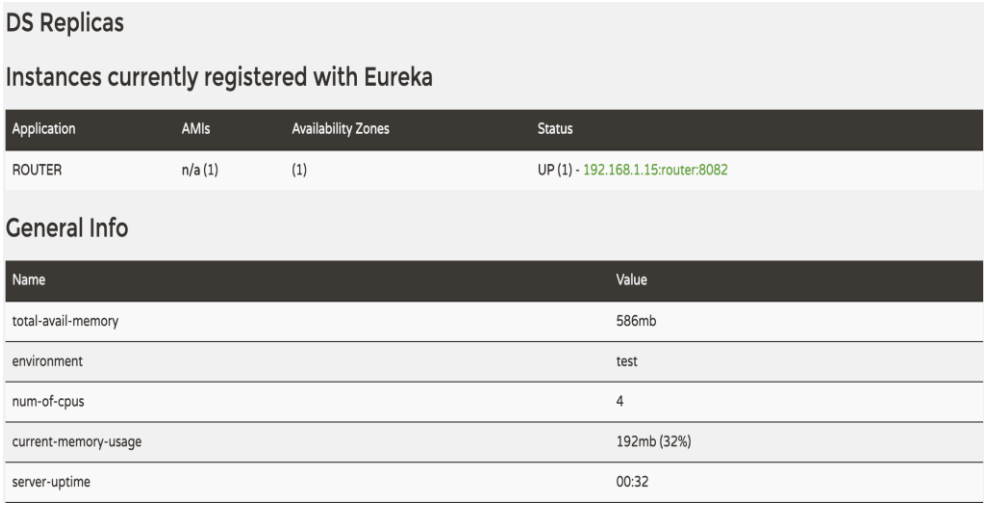


Fig. 68. Herramienta ZUUL(Router) como servicio.

Fuente: Elaborado por los Autores

Configuración de ZUUL (aplication.yml)

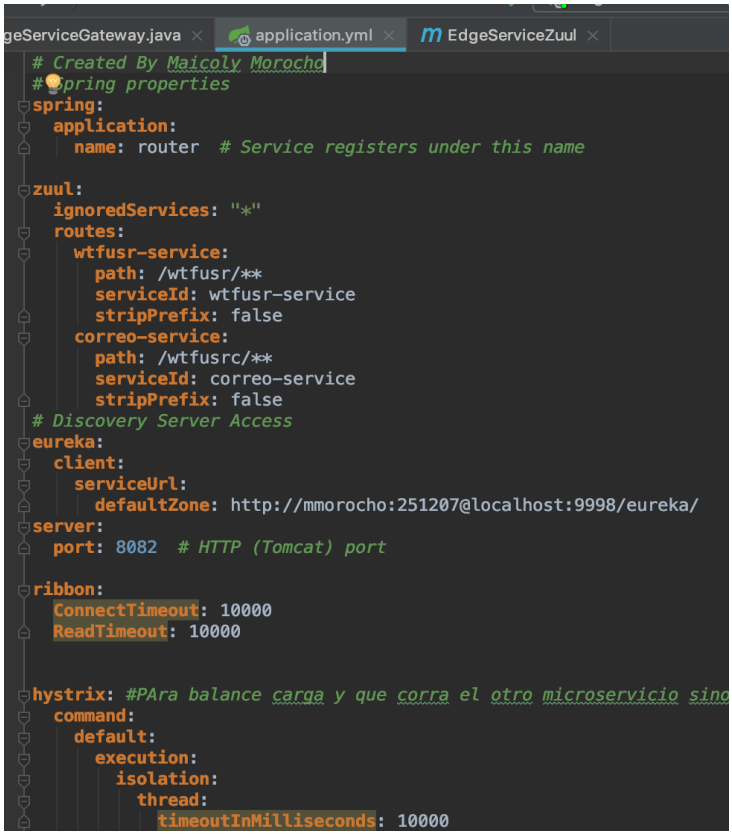


Fig. 69. Configuración del ZUUL (Router)
Fuente: Elaborado por los Autores

Microservicio ACCOUNTS-SERVICE

Este microservicio que se encarga de los mantenimientos o CRUD's de los principales servicios como registros, inicios de sesión, mantenimiento de cámara, etc., éste se encuentra registrado en el diccionario EUREKA, además se distingue desde que

máquina y puerto se están lanzando.

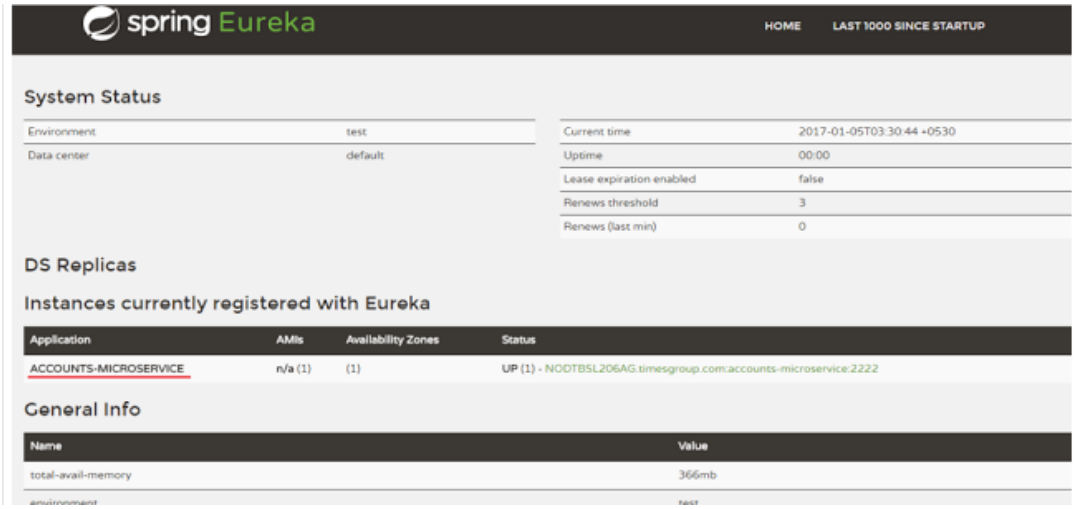


Fig. 70. servicio ACOOUN-MICROSERVICE.
Fuente: Elaborado por los Autores

Microservicio WTFUSR-SERVICE

EL microservicio encargado de las peticiones entrantes de la aplicación IOS/ANDROID y de la aplicación web, los principales servicios son:

- ✓ **Detectar rostro:** desde la aplicación móvil para subir un caso ya sea de secuestro o uno de los más buscados se debe revisar si la imagen tiene un rostro, una vez que llega el request, éste envía al servicio de inteligencia artificial para evaluar la imagen.
- ✓ **Fill Borders:** rescatamos desde la imagen dada el contorno del rostro o también rasgos faciales importantes como la boca u ojos.
- ✓ **Guardar Caso:** desde Typescript(IOS) se envía el JSON de todo el objeto CASO para guardar en el servidor central en estado revisión.
- ✓ **Despliegue:** una vez que un caso sea validado desde la página de administración, EUREKA enviará cada caso a los diferentes microservicios para que se actualicen en tiempo real.

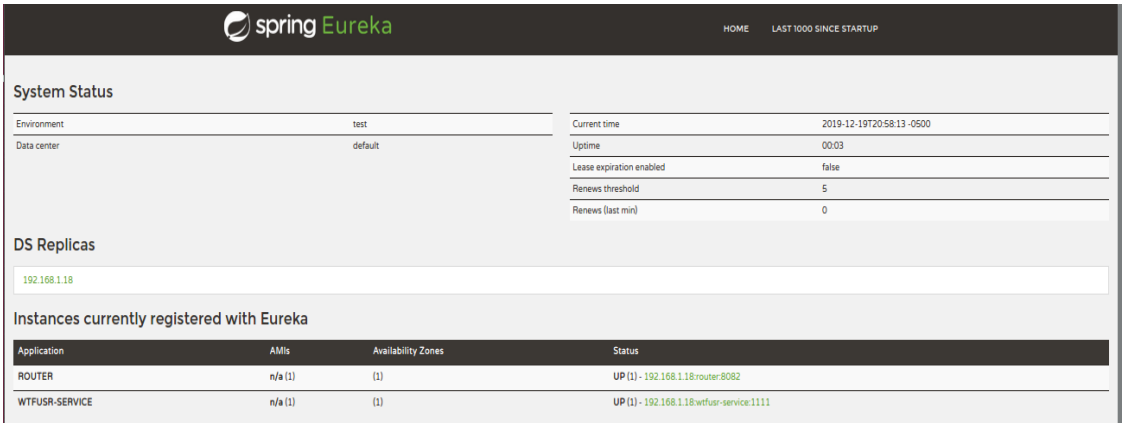


Fig 71. Servicio WTFUSR-SERVICE

Fuente: Elaborado por los Autores

Consumo del servicio SAVE-CASO

dentro del servicio WTFUSR-SERVICE existe el servicio para guardar un objeto CASO el cual contiene toda la información ya sea de un secuestro, más buscado, etc., este objeto nace desde una petición desde la aplicación móvil, llega al servicio de guardado, válida cierta información y será salvado en el core de base de datos principal, una vez que el administrador verifique que los datos sean legibles podrá activar o validar el caso para que se despliegue el rostro o imagen a todas las cámaras con sus servicios activos.

- ✓ Frontend móvil antes de enviar el caso
- ✓ Caso en el servidor de aplicaciones

HeartBeat a las cámaras

podemos enviar una señal conocida como heartbeat a las diferentes cámaras, desplegadas en los diferentes puntos remotos, para eso usamos también microservicios y el diccionario EUREKA para que nos diga la localización de ellos.

- Mapa de las cámaras desplegadas

Fig. 72. Mapa de cámaras desplegadas

Fuente: Elaborado por los Autores

- Test Conexión

Fig. 73. HeartBeat a las cámaras

Fuente: Elaborado por los Autores

Servicio de RE-Despliegue

en caso de que por alguna razón no se desplegó un rostro hacia las cámaras, podemos observar desde la administración el estado de cada caso, al estar TRUE el caso se

encuentra alojado en la cámara sino estará en FALSE, para en caso de estar en FALSE podemos usar el servicio de despliegue de un rostro a la cámara seleccionada.

- Servicio de RE-Despliegue

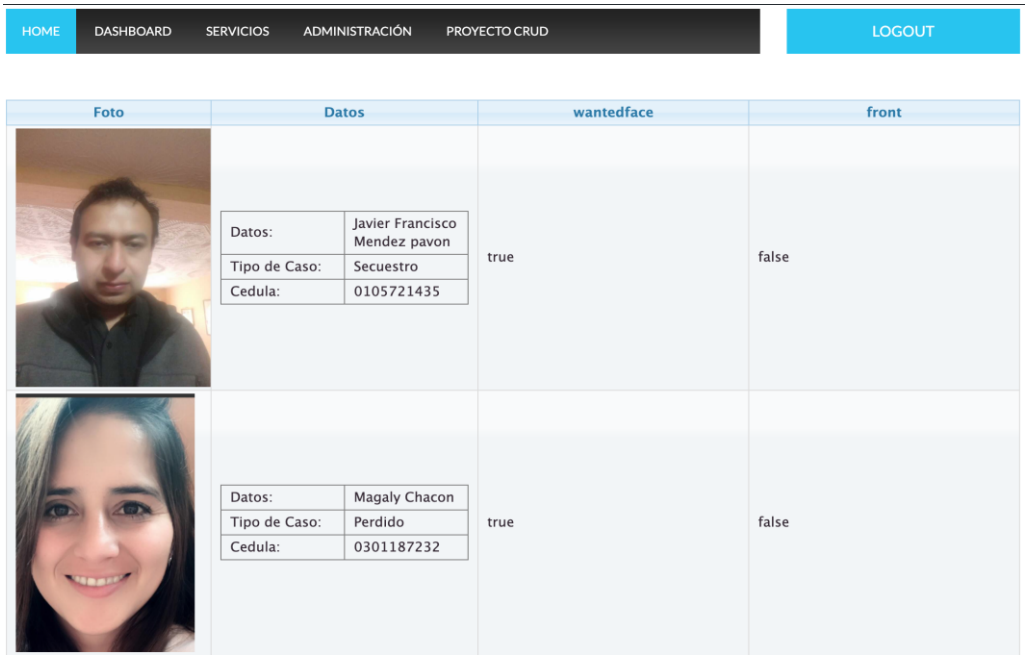


Fig. 74. Servicio de RE-Despliegue

Fuente: Elaborado por los Autores

Captura de rostros y notificaciones con sockets

para la captura de rostros cada servicio de inteligencia artificial se encuentra bajo una sesión usando Socket.IO ya que la comunicación es en tiempo real y mucho más rápida que REST, una vez que el servicio de inteligencia artificial detecte un rostro conocido, la sesión de socket inyectara al room o sala de cámaras alojada en JAVA el cuál comunicará a sus clientes JavaScript para mostrar notificaciones a los administradores que se encuentren haciendo seguimiento a través de las cámaras, además se envía SMS, emails de la detección del rostro.

Permite la comunicación entre el servidor y el cliente Java Script, en tiempo real, para una mejor

alternativa en lado del cliente en el tiempo de ejecución, en donde el servidor escuchara c el puerto 8080

En la (fig.,75) se creará en JavaScript la parte publica de la aplicación y comunicación

con el servidor.

```
<script type="text/javascript">
    var a=document.getElementById('h2:hd1').value;
    var sr=document.getElementById('h2:sr').value;
    var idp=document.getElementById('h2:idp').value;
    var dcc=document.getElementById('h2:dcc').value;
    // var url = 'ws://'+sr+':8080/wantedface/server/'+a+','+f,'+idp;
    var url = 'ws://192.168.1.10:8080/wantedface/server/'+a+','+'f,'+idp;
    var a=""
    console.log("URLF:",url);
    var webSocket =
        new WebSocket(url);
    webSocket.onerror = function (event) {
        onError(event)
    };
    webSocket.onopen = function (event) {
        onOpen(event)
    };
    webSocket.onmessage = function (event) {
        onMessage(event)
    };
    function onMessage(event) {
        // console.log("dcc: "+dcc);
        var temp=event.data.split(";")[0];
        // console.log("recibe "+event.data +" temp:"+temp+" a"+a);
        if(dcc==true || dcc=='true'){
            document.getElementById("f1:x").value = event.data;
            $("#f1\\:hdnBtn").click();
                if( a!=temp) {
                    a=temp
                    var x = document.getElementById("toast")
```

Fig. 75. Socket en java script

Fuente: Elaborado por los Autores

Sala de seguimiento de rostros y notificación en tiempo real

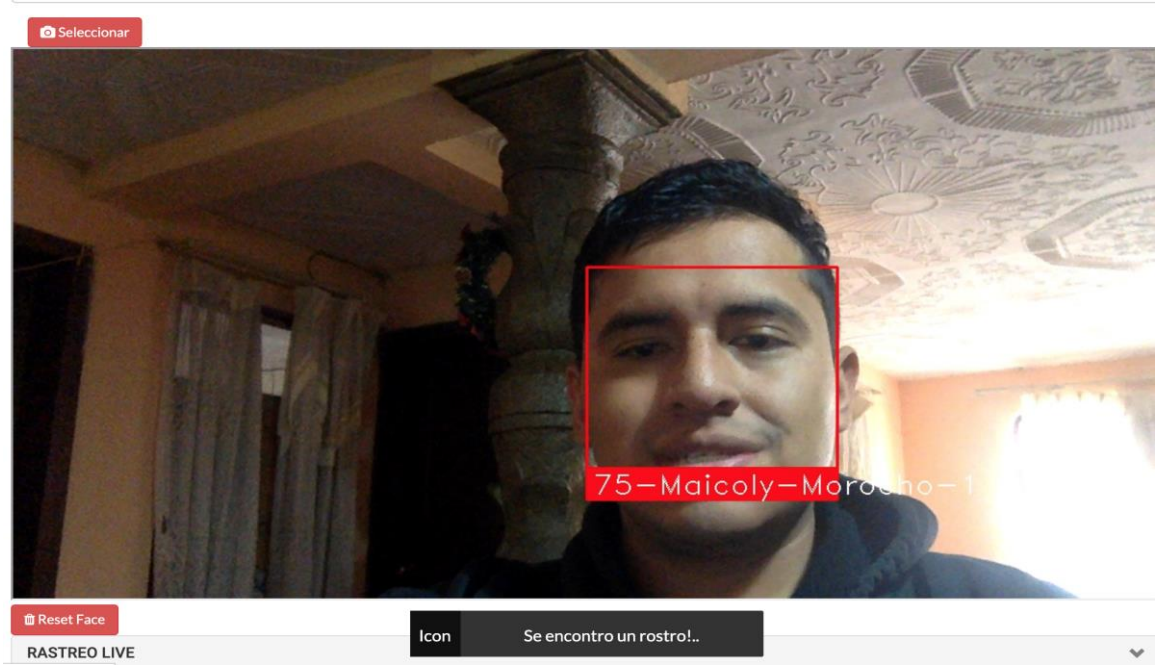


Fig. 76. Sala de seguimiento de rostros y notificación en tiempo real.

Fuente: Elaborado por los Autores

Notificaciones de rostros en tiempo real con su porcentaje de coincidencia y el tipo de caso, en este caso SECUESTRO

Fig. 77. Notificaciones de rostros en tiempo real con su porcentaje de conciencia y tipo de caso, en este caso SECUESTRO.

Fuente: Elaborado por los Autores

Historial de detección

Podemos observar las detecciones del rostro del Caso de Maicoly Morocho y demás información como la fecha detectada, dirección y el porcentaje de similitud.

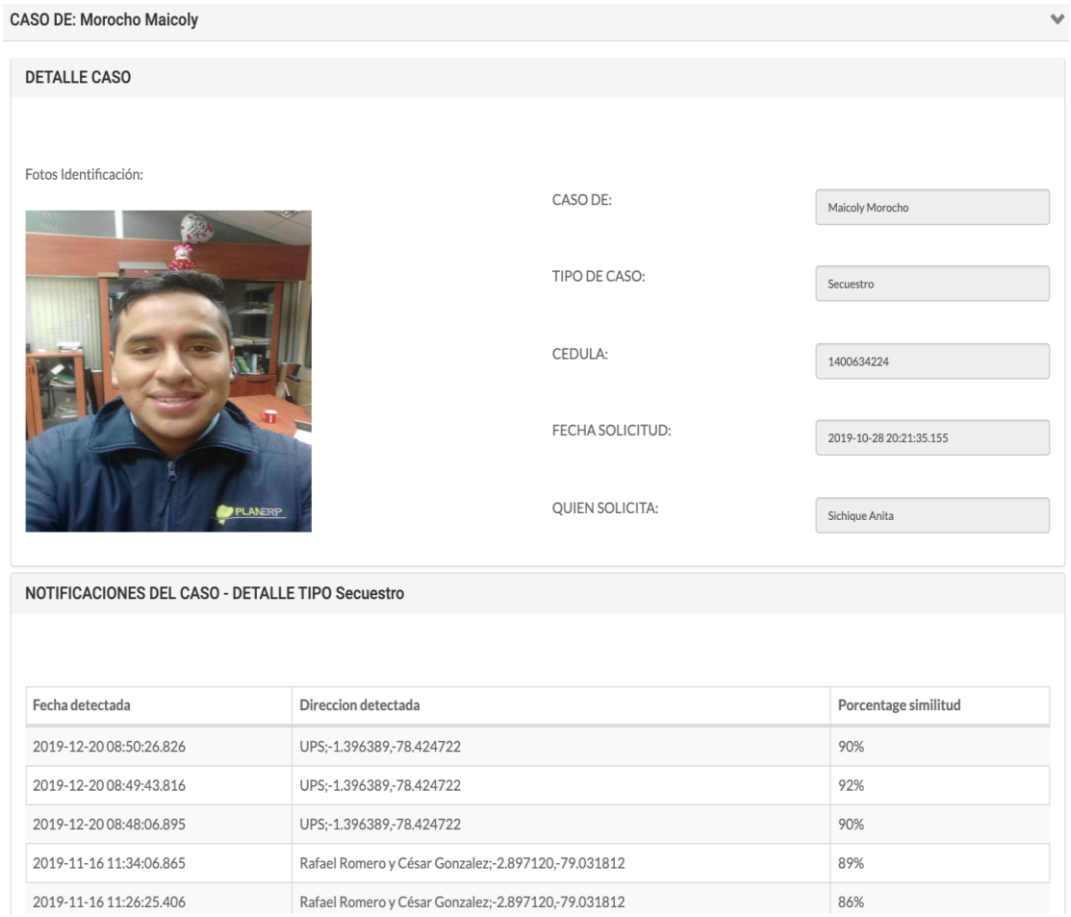


Fig. 78. Historial de detección.

Fuente: Elaborado por los Autores

8.6 Docker

En el apartado docker se ejecutará de manera independiente varios procesos y aplicaciones por separado haciendo un mejor uso de la infraestructura de docker donde se podrá ejecutar cualquier aplicación en distintos sistemas operativos, ayudara ahorrar recursos en componentes de hardware que estos están basado como máquinas virtuales, y tiempo.

Para la implementación se debe instalar y configurar las respectivas herramientas para el ambiente de docker como se indica a continuación, Anexo 3(Instalación docker).

Configuración e instalación de los paquetes, repositorios en Docker que soporte para Ubuntu 16.04.

Para proceder los pasos, se debe tener descargado el repositorio y ejecutar Docker, ingresar los siguientes comandos.

vps1@vps1:~$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D

vps1@vps1:~$ sudo apt-add-repository 'deb https://apt.dockerproject.org/repo ubuntu-xenial main'

vps1@vps1:~$sudo apt-get update

vps1@vps1:~$ apt-cache policy docker-engine

vps1@vps1:~$sudo apt-get install -y docker-engine
vps1@vps1:~$docker pull anitas/wantedface-subido
vps1@vps1:~$docker run -it anitas/wantedface-subido

Tabla de Paquetes requeridos para la implementación:

Descripción.	Paquetes.
Paquete para pip3 para python3	apt-get install python3-dev python3-pip
Para web service	Pip3 install flask
Para websocket	pip3 install websocket
Paquete para websocket cliente	pip3 install websocket-client

Tabla 15. Paquetes de instalación para Docker

Tabla de los comandos útiles:

Descripción.	Comandos.
Este sirve para descargar una imagen	*Docker pull nombre la imagen*
Muestra imágenes que tenemos descargadas	docker images
Muestra que contenedores están funcionando.	docker ps -a
Ejecuta la operación del contenedor basado en imagen	docker run -it *nombre imagen*
Muestra información de las imágenes, tamaño, fecha creación nombre.	docker info
Pública la imagen a la web.	docker push nombre imagen
Para la eliminación del docker.	docker rm nombre contenedor

Tabla 16. Comandos para Docker

Configuración de docker

Una vez terminado el proceso de la configuración e instalación se verificará si docker se instaló correctamente, comprobamos con el siguiente comando.

systemctl status docker.

En donde se trabajará con el docker debe actualizar los paquetes y comprar conectividad con el internet y descargar los paquetes necesarios.

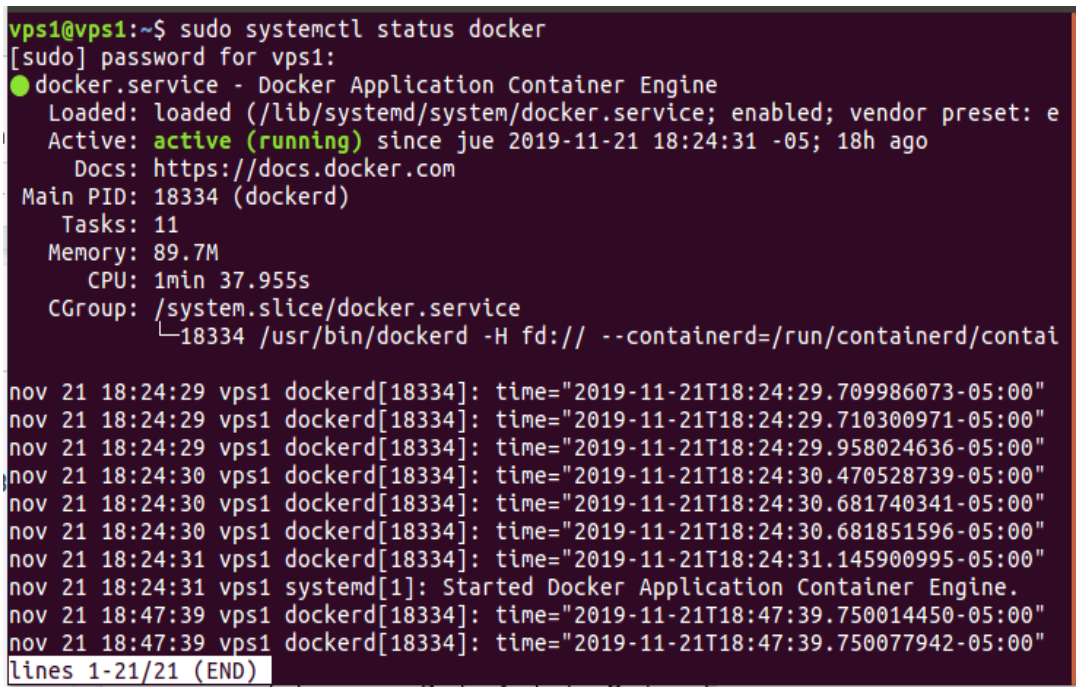


Fig. 79. Estado del docker

Fuente: Elaborado por los Autores

Conectividad de docker

Para la conexión de docker debe tener creado las credenciales que son el usuario y su contraseña una vez creado se ejecuta el siguiente comando ***docker login*** y esto lo pedirá el usuario y su contraseña una vez ingresado correctamente podrá acceder a ello.

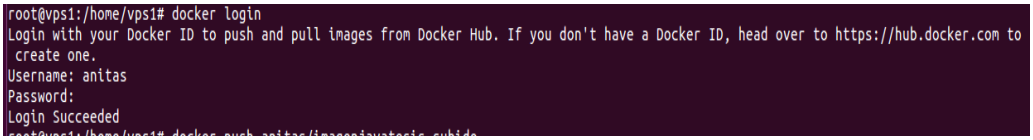


Fig. 80. Ingreso al docker mediante línea de comandos.

Fuente: Elaborado por los Autores

Publicación de imagen

Para realizar la publicación de la imagen al repositorio hub, se usará el siguiente comando para subir la imagen con el siguiente comando ***docker push nombreimgen*** durante el proceso se tardará unos minutos para que suba al repositorio. Como se verá en siguiente ejemplo el comando para subir

root@vps1:/home/vps1# docker push anitas/imagenjavatesis-subido

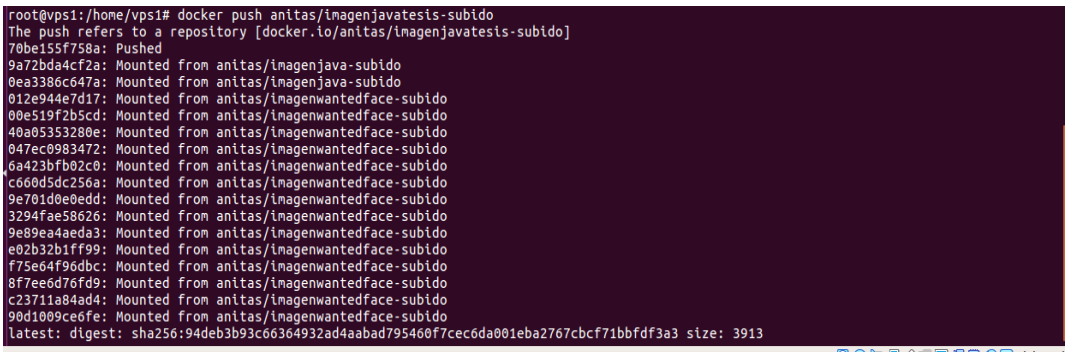


Fig. 81. Subiendo el docker al repositorio mediante línea de comandos.

Fuente: Elaborado por los Autores

Una vez terminado de subir el docker al repositorio web hub nos dirigimos a la sesión de docker ingresamos nuestro repositorio web, con nuestra credencial creados.

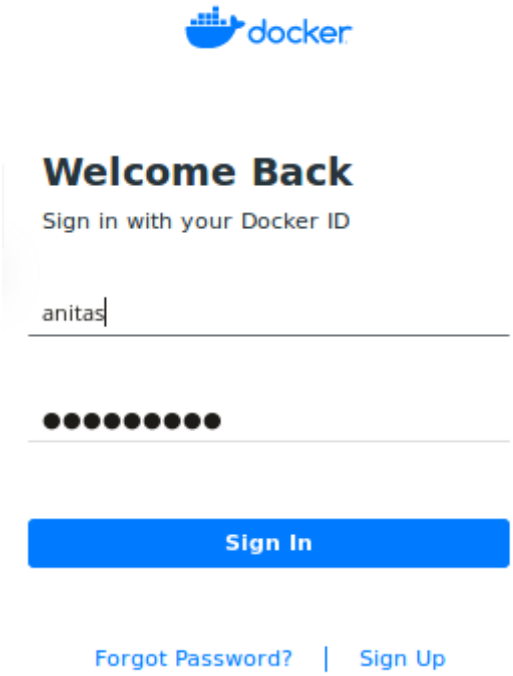


Fig. 82. Ingreso a la sesión docker.

Fuente: Elaborado por los Autores

Repositorios subidos en la web

Al terminar el proceso, se puede ingresar en la siguiente página *https://hub.docker.com/* y se verá que la imagen acaba de ser publicada.

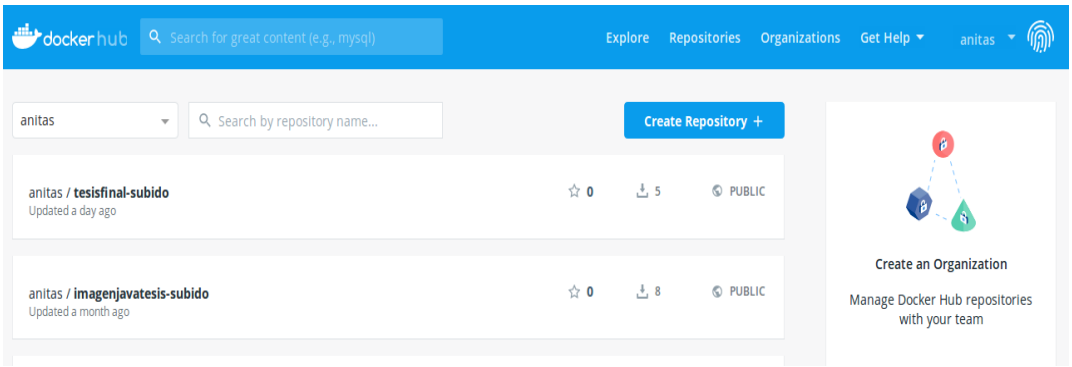


Fig. 83. Docker subiendo al repositorio web.

Fuente: Elaborado por los Autores

9. RESULTADOS

Los resultados se obtienen en base a 3 dispositivos que interactúan con los sistemas inteligentes, estos son: D-link Wireless N Day/Night Network Camera, cámara MacBook Pro 2014, NVidia Jetson Nano.

	D-link Wireless Network Camera	NVIDIA Jetson N ano	MacBook PRO
Demora: Valida los tiempos de respuesta desde que detecta un frame con rostros hasta que notifica al frontend administración	**Numero de Frames Capturados** =5 **Detección frame:**1582689952272 **Notificación:** 1582689952572 **Resultado** 0.3 seg.	**Numero de frames capturados** = 2 **Detección frame:** = 1582688489434 **Notificación:** 1582688489655 **Resultado** = 0.221 seg.	**Numero de frames capturados** = 7 **Detección frame** = 1582691199167 **Notificación:** 1582691199493 **Resultado:** 0.33 seg
Tráfico de red (Jnettop)	**Velocidad:** 50K/s **Consumo 1 clientes**: 0.5 KB/S	**Velocidad:** 1.73M/s **Consumo 1 clientes:** 1.6 MB/S	**Velocidad:** 1.3M/s **Consumo 1 clientes**: 1.5 MB/S
Porcentaje (%) con foto original	0.60-0.80%	0.85-0.95%	0.90-0.95%

Tabla 17. Elementos que influyente en la detección del rostro

Debemos tener en cuenta que influye algunos aspectos para la transferencia de un frame hacia el servidor conocido como demora, estos puntos pueden ser: tipo de conexión, tráfico de red, tipo de luminosidad, calidad de la cámara.

Mediante la gráfica siguiente podemos concluir que Nvidia Jetson Nano, tóma el menor tiempo desde la detección hasta la notificación de 221 milisegundos, esto influye la conexión directa hasta el servidor ya que la MacBook Pro se uso wifi.

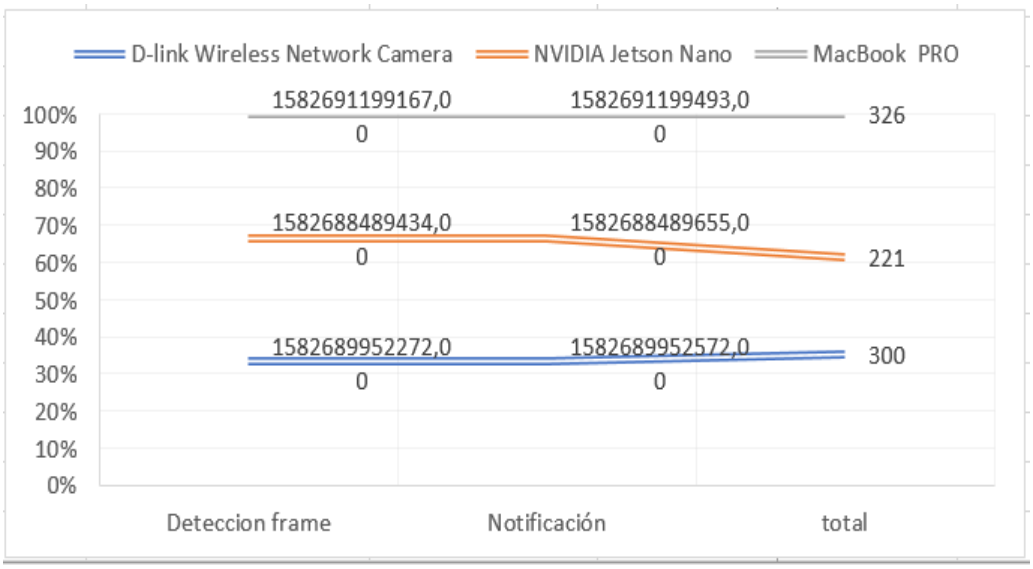


Fig. 84. Tiempo de demora para la notificación.

Fuente: Elaborado por los Autores

Por otro lado, tenemos la cantidad de frames con rostros capaz de detectar cada dispositivo, en este punto gana el de mayor calidad de cámara, distinguiendo la MacBook Pro y terminando con la D-Link.

Es decir, la MacBook Pro fue capaz de detectar 7 frames con rostros durante un tiempo de detección, mientras que la NVIDIA Jetson Nano 2 y 5 la D-Link.

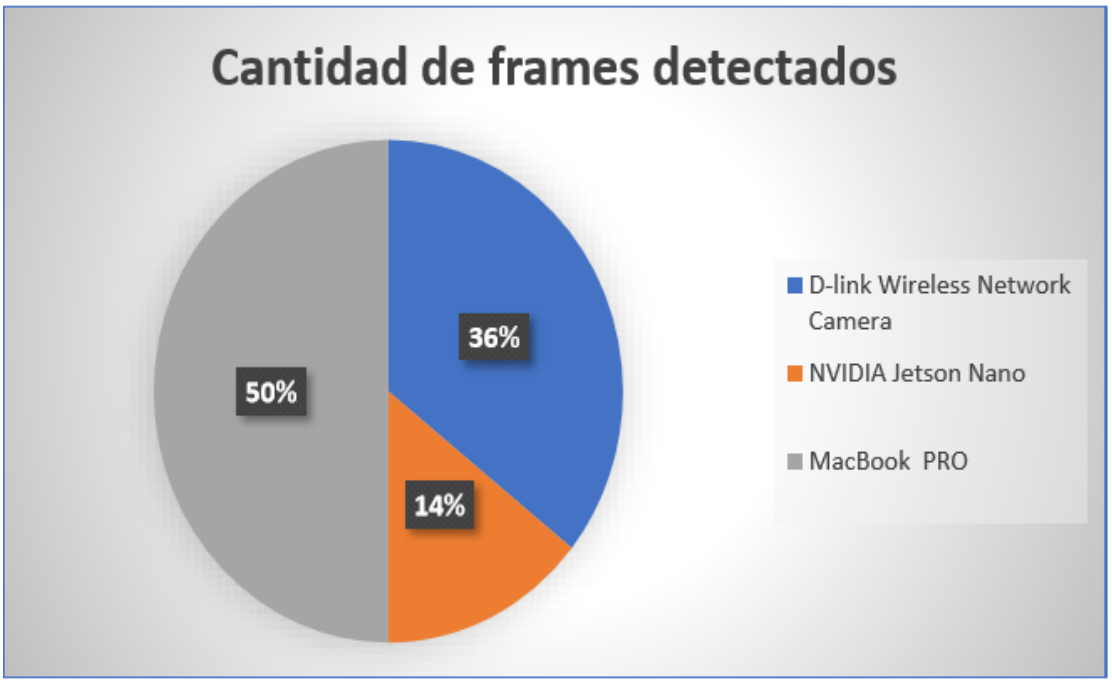


Fig. 85. Rostros detectados.

Fuente: Elaborado por los Autores

Elemento	D-link Wireless Network Camera	NVIDIA Jetson Nano	MacBook PRO
Memoria RAM	No proporciona	4.0GB/ 2 GB – SWAP 2GB	8GB / 6GB
CPU	No proporciona	ARMv8 Processor rev 1 (v8l)	Intel Core I7 3GHz
Núcleos	No proporciona	4	4
Frecuencia CPU	No proporciona	1479.0 MHz/Nucleo	2.1 GHz
Tamaño y nú mero de discos	No proporciona	1 disco 64GB	1 discos de 512 GB
Sistema Operativo	No proporciona	Ubuntu Server 16	MacOS Mojave

Tabla 18. Recursos de Hardware y Software

9.1 Análisis de la percepción de los resultados

Como pudimos observar en el apartado de resultados influye algunos aspectos para la

detección de un rostro, entre ellos esta la luminosidad, tráfico de red, calidad de la cámara, etc.

Lo que nuestro sistema y requerimientos necesita es rapidez ante una detección, por lo cuál NVIDIA Jetson Nano es la mejor opción ya que hace eso de sus recursos de manera responsable pudiendo en 0,221 segundos detectar y notificar al sistema de una detección con una precisión que va desde el 0.85%-0.95%, con un consumo de 1.46 Ghz por núcleo de su CPU.

10. EVALUACION DE LOS RESULTADOS

10.1 Análisis de la percepción por parte de los usuarios al ver el sistema

El sistema de reconocimiento facial es una propuesta de ayuda a la sociedad ecuatoriana, para brindar mayor seguridad en los medios de transporte público que son de uso frecuente en las diferentes ciudades, por lo que se ha previsto realizar una encuesta para estar al tanto de las opiniones de las personas particulares que usan los servicios de movilidad y cuál sería su reacción o acogida a una aplicación de seguridad hoy en día. (Referencia Encuesta)

10.2 Análisis de la percepción de los resultados en gráficos por pregunta

La ciudad de Cuenca es la capital del Azuay, es conocida como Santa Ana de los cuatro ríos, se encuentra situada al sur del país del Ecuador con una extensión de 8.310 km² y sus límites son al norte con Cañar, al sur con Loja, por el occidente con Guayas, al suroccidente con El Oro, al este con Morona Santiago y al sureste con Zamora Chinchipe, esta ciudad atraviesa las montañas andinas del país [42], sus llamativos y atractivos turísticos por sus artesanías como es el sombrero de paja toquilla, esto hace que muchos turistas extranjeros y nacionales que visiten a esta ciudad. Así como también hace que estudiantes migren a esta ciudad desde otras provincias a realizar sus estudios.

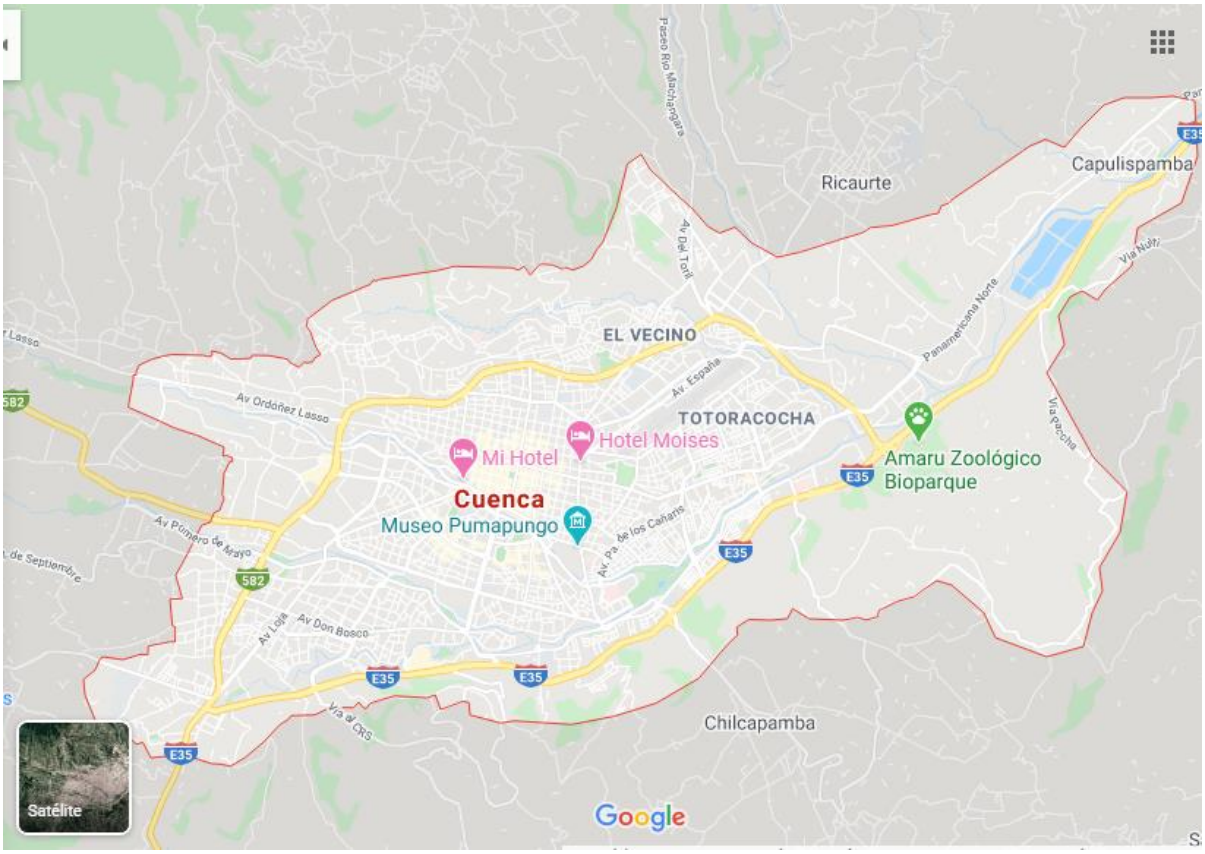
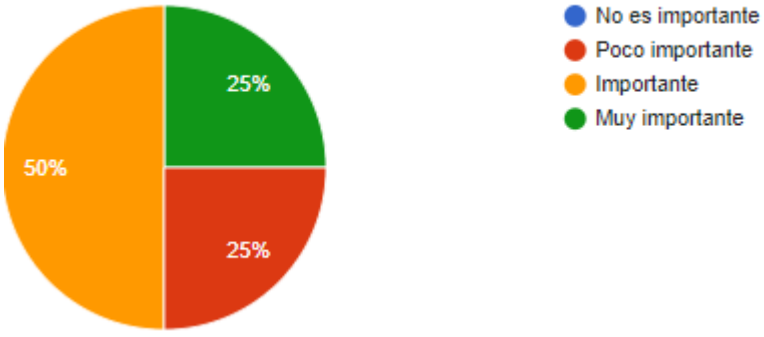


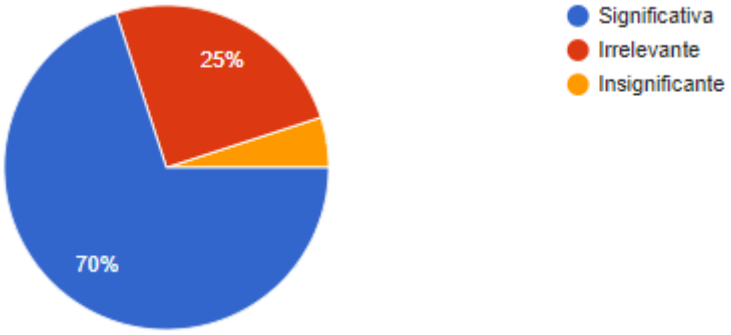
Fig. 86. Mapa de la provincia del Azuay ciudad de cuenca.

Se ha visto a base de esto se planteó realizar una encueta a la población cuenca, la evaluación correspondiente a este sistema se ha aplicado a diferentes personas de la ciudad de Cuenca para verificar su utilidad en la vida diaria con un número aproximado de 100 personas que se ha evaluado para realizar el proceso de este proyecto técnico mediante una.

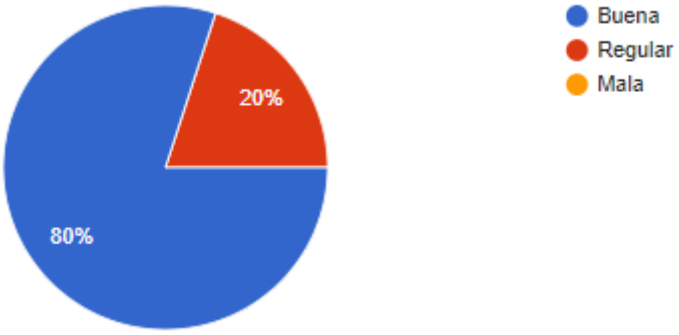
1. ¿Qué tan importante considera la implementación de la aplicación dentro los medios trasporte público?



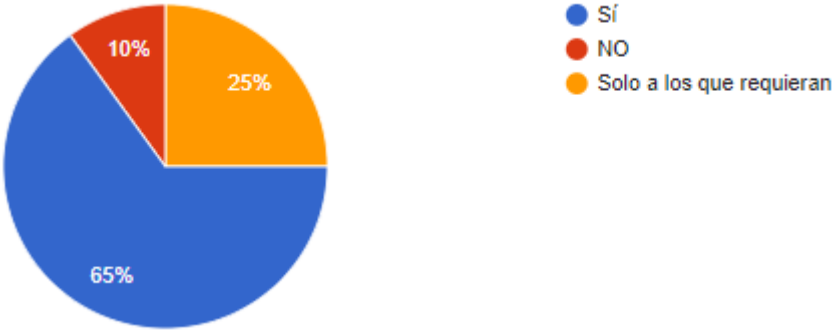
2. ¿La utilización de esta aplicación será de gran contribución para la seguridad de la ciudadanía?



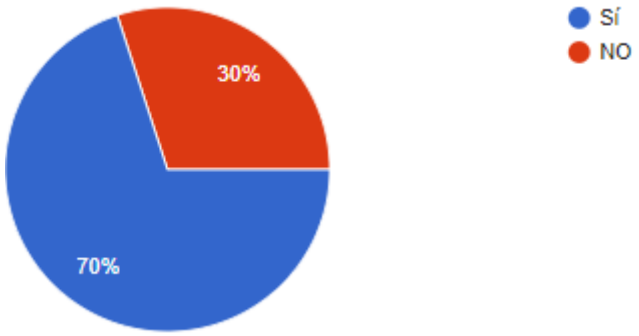
3. ¿Siente confianza al saber que se implementará una aplicación en los medios de transporte que ayude a la detección de los delincuentes y personas extraviadas?



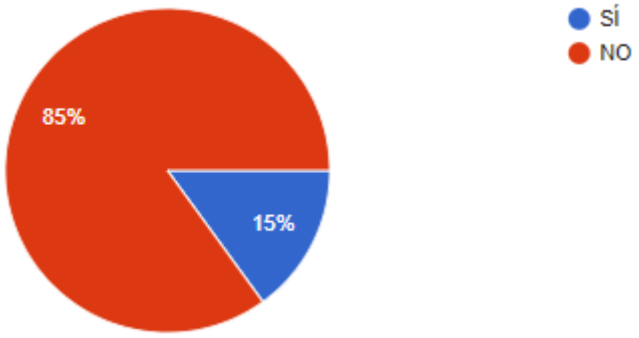
4. ¿La aplicación será de gran ayuda a la ciudadanía para poder hacer sus denuncias con gran facilidad?



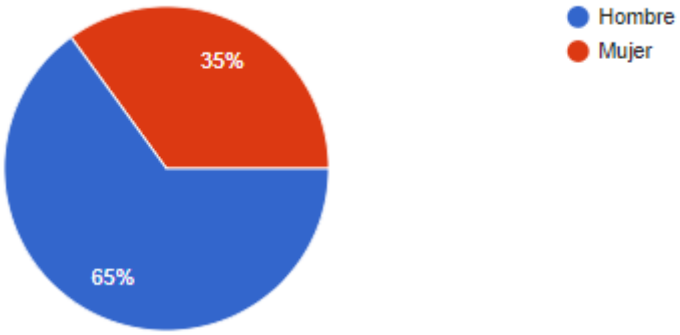
5. ¿Le interesaría utilizar esta aplicación en su celular para estar alerta?



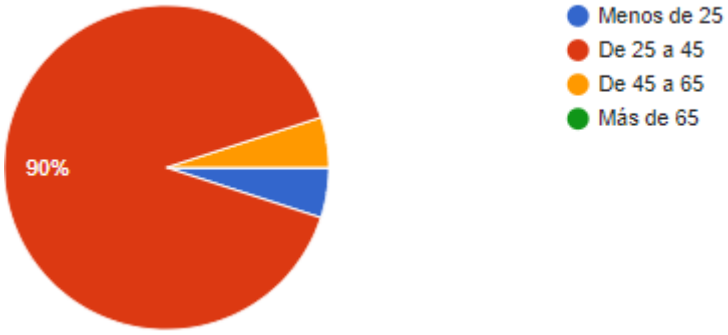
6. ¿Ha escuchado recientemente otra aplicación que cumplan estas características dentro del mercado?



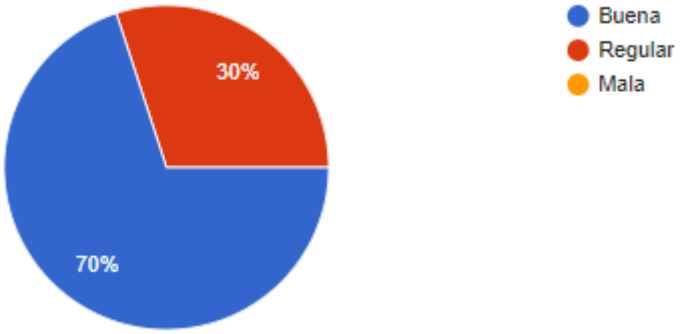
7. ¿Podría decirme su género?



8. ¿Su edad?



9. ¿Cómo calificarías la confiabilidad de la aplicación y lo recomendaría a algún amigo/a o miembro de familia?



11.CONCLUSIONES

Gracias al uso de las tecnologías de la información y comunicaciones (TIC's), es una herramienta primordial dentro de la seguridad, el proyecto técnico está enfocado a la seguridad e integridad de la ciudadanía. Dentro del proceso de desarrollo se ve la importancia de la inteligencia artificial, ya que, dentro de esta área, tiene la finalidad el reconocimiento facial en la arquitectura de microservicios, gracias a esta tecnología, se pudo hacer cumplimiento del sistema de reconocimiento facial de los prófugos por la ley.

La complejidad crece con el aumento de microservicios, es decir a mayores servicios que requiera nuestra aplicación mayor será la fase de implementación como dar su respectivo soporte, para solventar este tipo de inconveniente se puede empaquetar los servicios base, como los futuros, en contenedores en este caso docker.

Probar una aplicación en microservicios como la transaccionalidad requiere de más esfuerzo que una tradicional, una vez que se tenga un conocimiento maduro la implementación y testeo es más fácil.

12. RECOMENDACIONES

De haber culminado el proyecto de titulación se ve algunos aspectos relevantes para quien estén interesados a reutilizar o dar seguimiento del proyecto se plantea algunas recomendaciones que ve a lo siguiente.

Con respecto a este tipo de proyecto no es aconsejable usar Open Nébula y Docker, porque todos los servicios que usan Inteligencia Artificial consumen mucha CPU, GPU, RAM por lo tanto Open Nébula y Dockers están demás para este tipo de proyecto, con una excepción, los contenedores Docker se pueden usar para empaquetar todo el proyecto de reconocimiento facial y brindar a un usuario externo para que pueda usarlo por ejemplo en su hogar o alguna empresa y gestionar la seguridad como alternativa.

Al respecto al servidor de Open Nébula sobre el sistema operativo Ubuntu 18.04 server y 16.04 es aconsejable utilizar la versión 5.8 ya que la versión anterior tiene errores como **scp: error: unexpected filename** durante la instalación y configuración.

13. REFERENCIAS

[1] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, Sateesh Addepalli. Fog Computing and Its Role in the Internet of Things

[2] Docker. Disponible online: https://docs.docker.com (accedido en 01 de Marzo de 2020).

[3] LÓPEZ, Daniel, et al. Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web. 2017.

[4] Newman, S. Building Microservices. O'Reilly Media, Inc. (2015).

[5] Fowler, M.; Lewis, J. Microservices. Viittattu, (2014).

[6] Toledano López, López del Castillo, & Vázquez Sanchez, 2018.

[7] PONSICO MARTIN, Pol. Tecnología de Contenedores Docker. 2017. Tesis de Licenciatura. Universitat Politècnica de Catalunya.

[8] ROJAS CARDENAS, Miguel Abdon. CONTENEDORES DOCKER Y EL PROCESO DE DESARROLLO DE SOFTWARE. 2019. Tesis Doctoral.

[9] MONTORO, Arturo Fernández. Python 3 al descubierto. RC Libros, 2012.

[10] MADERA, Yaguapaz; GONZALO, Luis. Estudio del framework IONIC 2 para el desarrollo de aplicaciones móviles hibridas. 2018. Tesis de Licenciatura.

[11] MATUTE SÁNCHEZ, Martha Liliana; CONTRERAS ALVARADO, Carlos Rafael. Diseño y desarrollo de un asistente robótico basado en sistemas embebidos y aplicaciones móviles

como herramienta de soporte pedagógica para niños de uno a cinco años. 2019. Tesis de Licenciatura.

[12] LOCKHART, Thomas. Tutorial de PostgreSQL. El equipo de desarrollo de PostgreSQL, 1988, vol. 10.

[13] J. M. Meré, «, Técnicas de Visión por Computador para la Reconstrucción en Tiempo Real de la forma 3D de Productos Laminados. Gijón: Universidad de Oviedo,» 2008. [En línea]. Available: http://www.tesisenred.net/bitstream/handle/10803/11139/UOV0060TJMM.pdf?seque nce=1.

[14] QUISPE, Rinna Ximena Chipana. Cloud Computing para Aplicaciones Logísticas. *Rev. Inf. Tecnol. Soc*, 2012, p. 30.

[15] RODRIGUEZ, Ismael Pablo, et al. Despliegue de un Cloud Privado para entornos de cómputo científico. En *XVII Congreso Argentino de Ciencias de la Computación*. 2011.

[16] NAVARRO, J. Arévalo. *Cloud Computing: fundamentos, diseño y arquitectura aplicados a un caso de estudio*. 2011. Tesis Doctoral. Tesis de Fin de Máster. Máster Oficial en Tecnologías de la Información y Sistemas Informáticos. Universidad Rey Juan Carlos. 2011. Fecha de consulta: 10 de Abril de 2015. URL: URL: http://eficiencia. urjc. es/bitstream/10115/5945/1/MemoriaTFMFINAL-2. pdf.

[17] KUMAR, Rakesh, et al. Open source solution for cloud computing platform using OpenStack. *International Journal of Computer Science and Mobile Computing*, 2014, vol. 3, no 5, p. 89-98.

[18] DE LA TORRE, María Cabrera Gómez. GESTIÓN DE CONTENEDORES DOCKER.

[19] F. M. d. P. A. P. F. A. M. C. J. O. a. E. V. A. González, «Técnicas y Algoritmos Básicos de Visión Artificial,» Universidad de La Rioja, Servicio de Publicaciones, 2006. [En línea]. Available: https://publicaciones.unirioja.es/catalogo/online/VisionArtificial.pdf.

[20] J. M. Meré, «, Técnicas de Visión por Computador para la Reconstrucción en Tiempo Real de la forma 3D de Productos Laminados. Gijón: Universidad de Oviedo,» 2008. [En línea]. Available:

http://www.tesisenred.net/bitstream/handle/10803/11139/UOV0060TJMM.pdf?sequence=1.

[21] F. M. d. P. A. P. F. A. M. C. J. O. a. E. V. A. González, «Técnicas y Algoritmos Básicos de Visión Artificial,» Universidad de La Rioja, Servicio de Publicaciones, [En línea]. Available:: https://publicaciones.unirioja.es/catalogo/online/VisionArtificial.pdf.

[22] SANABRIA, John J.; ARCHILA, John F. Detección y análisis de movimiento usando visión artificial. *Scientia et technica*, 2011, vol. 16, no 49, p. 180-188.

[23] CABALLERO BARRIGA, Edison Rene, et al. Aplicación práctica de la visión artificial para el reconocimiento de rostros en una imagen, utilizando redes neuronales y algoritmos de reconocimiento de objetos de la biblioteca opencv. 2017.

[24] CADAVID, Andrés Navarro; MARTÍNEZ, Juan Daniel Fernández; VÉLEZ, Jonathan Morales. Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, 2013, vol. 11, no 2, p. 30-39.

[25] DIMES, Troy. *Conceptos Básicos de Scrum: Desarrollo de software Agile y manejo de proyectos Agile*. Babelcube Inc., 2015.

[26] TOMASELLI, Gabriela Patricia, et al. SCRUM: Una revisión de la literatura. 2013.

[27] CASO, N. SCRUM development process. *Universidad Tecnológica Nacional.(En línea). Consultado*, 2004, vol. 22.

[28] I. T. S. S.L., «Cloud Computing,» 2014. [En línea]. Available: https://www.ittravelservices.com/cloud-computing.html. [Último acceso: 2020].

[29] OpenFogComputing, OpenFog Reference Architecture for Fog Computing, 2017.

[30] Pandorafms, «Monitorizacion: retos derivados de la naturaleza de los contenedores,» 21 02 2018. [En línea]. Available: https://pandorafms.com/blog/es/monitorizacion-docker/. [Último acceso: 2020].

[31] ARORA, Rachna; PARASHAR, Anshu; TRANSFORMING, Cloud Computing Is. Secure user data in cloud computing using encryption algorithms. *International journal of engineering research and applications*, 2013, vol. 3, no 4, p. 1922-1926.

[32] KUMAR, Rakesh, et al. OpenNebula: Open Source IaaS Cloud Computing Software Platforms. En *National Conference on Computational and Mathematical Sciences (COMPUTATIA-IV), Technically Sponsored By: ISITA and RAOPS, Jaipur*. 2014.

[33] SOLA CARABALLO, Carlos de. Explotación de OpenNebula como plataforma cloud IaaS para la docencia. 2015.

[34] PELUFFO, O. Diseño de telefonía IP a través del Cloud Computing bajo una plataforma de software libre OpenNebula.

[34] LÓPEZ LENGUA, Eliana Margarita. Sistema de evaluación virtual. 2011.

[35] OpenNebula, «OpenNebula,» 2019. [En línea]. Available: https://docs.opennebula.org/5.10/deployment/open_cloud_host_setup/kvm_driver.html. [Último acceso: 2020].

[36] OpenNebula, «OpenNebula,» 2017. [En línea]. Available: https://hackaday.io/project/10529-solar-powered-cloud-computing/log/63777-opennebula-front-end-setup. [Último acceso: 2020].

[37] VALVERDE GALÁN, Paola Adriana; CALLE ROMERO, Paúl Esteban. *CurSAE-V: Sistema de ambientes de estudio y escritorios virtuales basados en la nube*. 2018. Tesis de Licenciatura.

ANEXOS

Anexo 1: Encuesta Realizada.

ESTA ENCUESTA SOBRE EL RECONOCIMIENTO FACIAL Y UNA APLICACIÓN MÓVIL PARA EL MEJORAMIENTO DE LA SEGURIDAD EN LA CIUDAD DE CUENCA.

Esta encuesta tiene por objetivo determinar la percepción de la seguridad en la ciudadanía de la ciudad de Cuenca - Ecuador respecto al reconocimiento facial de las personas en el transporte público y la aplicación móvil, cuya finalidad de regenerar la confianza en la ciudadanía

Más o menos ponle como le parece importante esta aplicación

1. ¿Qué tan importante considera la implementación de la aplicación dentro los medios trasporte público?

   ☐ No es importante
   ☐ Poco importante
   ☐ Importante
   ☐ Muy importante

2. ¿La utilización de esta aplicación será de gran contribución para la seguridad de la ciudadanía?

   ☐ Significativa
   ☐ Irrelevante
   ☐ Insignificante

3. ¿Siente confianza al saber que se implementará una aplicación en los medios de transporte que ayude a la detección de los delincuentes y personas extraviadas?

   ☐ Buena
   ☐ Regular
   ☐ Mala

4. ¿La aplicación será de gran ayuda a la ciudadanía para poder hacer sus denuncias con gran facilidad?
   ☐ Sí NO ☐
   ☐ Solo a los que requieran

5. ¿Le interesaría utilizar esta aplicación en su celular para estar alerta?
   SÍ ☐ NO ☐

6. ¿Ha escuchado recientemente otra aplicación que cumplan estas características dentro del mercado?
   ☐ SÍ NO ☐

7. ¿Podría decirme su género?

   ☐ Hombre ☐ Mujer

8. ¿Su edad?

   ☐ Menos de 25
   ☐ De 25 a 45
   ☐ De 45 a 65
   ☐ Más de 65

9. ¿Te gusta la Aplicación de reconocimiento facial, lo recomendaría a algún amig@ o miembro de familia ?:

   ☐ SÍ

   ☐ NO

La encuesta ha terminado
Muchas Gracias por su colaboración

Anexo 2: Manual OpenNebula.

INSTALACION OPENNEBULA CON LA VERSION 5.8

Durante la instalación del software Opennebula, este se divide en dos partes, los cuales son Frontend, Node (KVN), el Frontend este representa la interfaz gráfica (GUI) y el Node en este se designa los recursos para las máquinas virtuales.

Configuración Preliminar

Para implementación se utilizó dos máquinas GNU/Linux Ubuntu, se debe instalar varios paquetes para ejecución el cual será para su funcionalidad.

Actualización de los repositorios

sudo apt-get update

sudo apt-get upgrade

Configuración del adaptador puente en el FRONT-END

Configuramos el host y la IP para la implementación, se utiliza el siguiente comando: *sudo nano /etc/hosts*

```
 GNU nano 2.5.3                File: /etc/hosts                      Modified

127.0.0.1       localhost
127.0.1.1       front
192.168.1.9     front
192.168.1.10    wantedface

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Captura 1. Configuración del Host

Agregar repositorio

Para la agregación del repositorio de epel y OpenNebula se manda a ejecutar la siguiente línea Proceso de instalación de Opennebula 5.8 FROND on Ubuntu 16.04, se

ejecuta como root:

sudo su wget -q -O- https://downloads.opennebula.org/repo/repo.key | apt-key add -

```
root@front:/home/maestro# wget -q -O- https://downloads.opennebula.org/repo/repo
.key | sudo apt-key add -
OK
root@front:/home/maestro#
```

Captura 2. Agregación del repositorio

Agregación del repositorio al sistema

Con esta línea de comando se agrega repositorio al sistema

echo "deb https://downloads.opennebula.org/repo/5.8/Ubuntu/16.04 stable opennebula" > /etc/apt/sources.list.d/opennebula.list

```
root@front:/home/maestro# echo "deb https://downloads.opennebula.org/repo/5.8/Ub
untu/16.04 stable opennebula" > /etc/apt/sources.list.d/opennebula.list
root@front:/home/maestro#
```

Captura 3. Agregación del repositorio al sistema

Instalación de la base de datos

Agregación del repositorio al sistema para Base de datos, con estos comandos se importa la clave PGP.

sudo apt -y install software-properties-common dirmngr

```
root@front:/home/maestro# sudo apt -y install software-properties-common dirmngr
Reading package lists... Done
Building dependency tree
Reading state information... Done
dirmngr is already the newest version (2.1.11-6ubuntu2.1).
software-properties-common is already the newest version (0.96.20.9).
The following package was automatically installed and is no longer required:
  snapd-login-service
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 18 not upgraded.
```

Captura 4. Agregación del repositorio PGP al sistema

sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80

0xF1656F24C74CD1D8

```
root@front:/home/maestro# sudo apt-key adv --recv-keys --keyserver hkp://keyserv
er.ubuntu.com:80 0xF1656F24C74CD1D8
Executing: /tmp/tmp.Z7x8DdRI9b/gpg.1.sh --recv-keys
--keyserver
hkp://keyserver.ubuntu.com:80
0xF1656F24C74CD1D8
gpg: requesting key C74CD1D8 from hkp server keyserver.ubuntu.com
gpg: key C74CD1D8: public key "MariaDB Signing Key <signing-key@mariadb.org>" im
ported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
root@front:/home/maestro#
```

Captura 5. Repositorio al sistema

sudo add-apt-repository 'deb [arch=amd64] http://mirror.zol.co.zw/mariadb/repo/10.3/ubuntu xenial main'

```
root@front:/home/maestro# sudo add-apt-repository 'deb [arch=amd64] http://mirro
r.zol.co.zw/mariadb/repo/10.3/ubuntu xenial main'
root@front:/home/maestro#
```

Captura 6. Repositorio al sistema

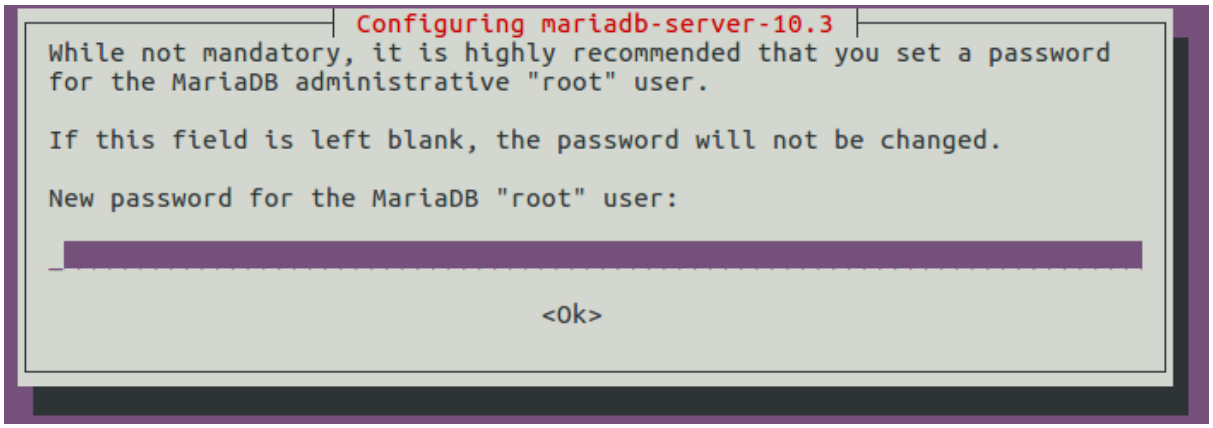
Se actualiza la lista de paquetes del sistema e instalar Base de Batos

sudo apt update

sudo apt -y install mariadb-server mariadb-client

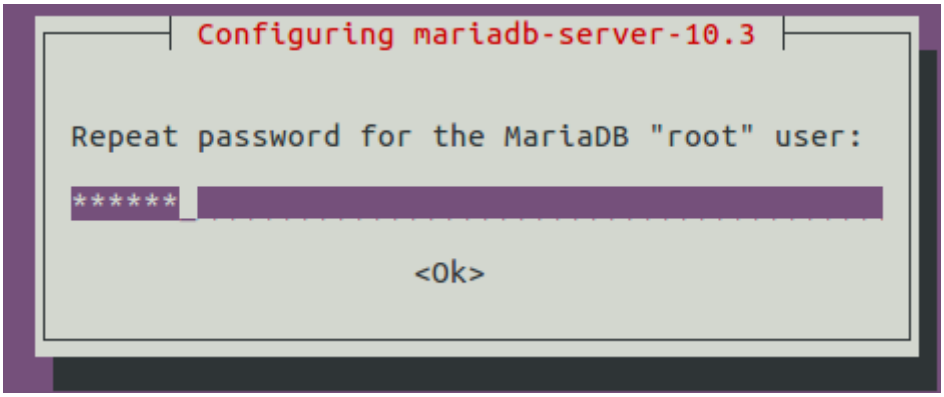
En esta lo pedirá la contraseña de root de Base de Datos

Se ingresa una contraseña para establecer.



Captura 7. Ingreso de la contraseña para BD

Confirmar la contraseña y presionar OK para la configuración de la contraseña y se puede ver la confirmación de la Base de Datos



Captura 8. confirmación de la clave para BD

CONFIGURACION DE BASE DE DATOS

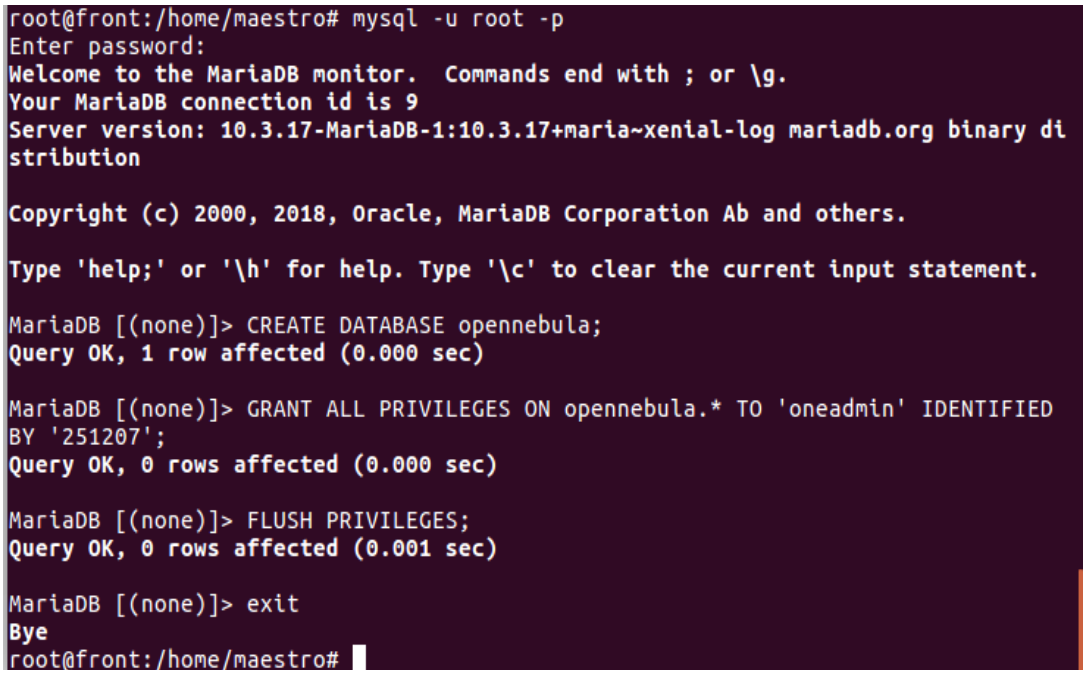
Se puede configuración de la versión de la base de datos instalada y se puede iniciar con el usuario root y procede crear una base de datos, y a esta se da todos los privilegios

mysql -u root -p

CREATE DATABASE opennebula;

GRANT ALL PRIVILEGES ON opennebula. TO 'oneadmin' IDENTIFIED BY '251207';*

FLUSH PRIVILEGES;



Captura 9. Creación de la BD

Install OpenNebula Front-end package

apt-get update

apt-get install opennebula opennebula-sunstone opennebula-gate opennebula-flow

Se procede la Instalación de los paquetes que necesita la biblioteca Ruby, el cual OpenNebula suministra un sript que instale lo necesario

sudo /usr/share/one/install_gems

```
root@front:/home/maestro# sudo /usr/share/one/install_gems
Fetching: bundler-2.0.2.gem (100%)
Successfully installed bundler-2.0.2
Parsing documentation for bundler-2.0.2
Installing ri documentation for bundler-2.0.2
Done installing documentation for bundler after 4 seconds
1 gem installed
Distribution "debian" detected.
About to install these dependencies:
* gcc
* rake
* libxml2-dev
* libxslt1-dev
* patch
* g++
* libsqlite3-dev
* libcurl4-openssl-dev
* libmysqlclient-dev
* ruby-dev
* make

Press enter to continue...
```

Captura 10. Script que necesita la biblioteca Ruby

```
Installing net-ldap 0.16.1
Fetching ox 2.10.0
Installing ox 2.10.0 with native extensions
Using parse-cron 0.1.4
Fetching polyglot 0.3.5
Installing polyglot 0.3.5
Fetching rack 2.0.6
Installing rack 2.0.6
Fetching rack-protection 2.0.5
Installing rack-protection 2.0.5
Fetching sequel 5.17.0
Installing sequel 5.17.0
Fetching tilt 2.0.9
Installing tilt 2.0.9
Fetching sinatra 2.0.5
Installing sinatra 2.0.5
Fetching sqlite3 1.3.13
Installing sqlite3 1.3.13 with native extensions
Fetching thin 1.7.2
Installing thin 1.7.2 with native extensions
Fetching treetop 1.6.10
Installing treetop 1.6.10
Fetching trollop 2.9.9
Installing trollop 2.9.9
Using uuidtools 2.1.5
Fetching zendesk_api 1.16.0
Installing zendesk_api 1.16.0
Bundle complete! 27 Gemfile dependencies, 53 gems now installed.
Use `bundle info [gemname]` to see where a bundled gem is installed.
Post-install message from trollop:
!    The 'trollop' gem has been deprecated and has been replaced by 'optimist'.
!    See: https://rubygems.org/gems/optimist
!    And: https://github.com/ManageIQ/optimist
root@front:/home/maestro#
```

Captura 11. Paquetes instalados

Presionamos Enter para proseguir la instalación y una vez terminado

Configure OpenNebula DB

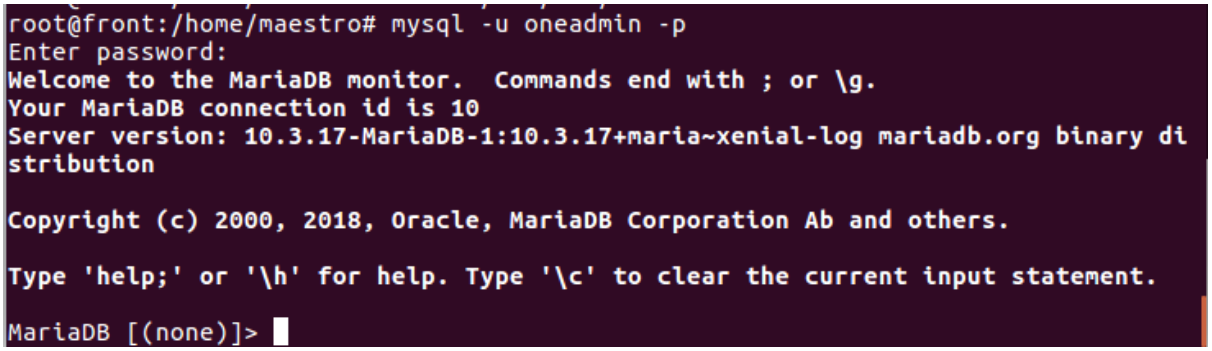
Para la configuración se abre el archivo oned.conf y para la configuración de la base de Datos como vera la siguiente imagen.

nano /etc/one/oned.conf

Captura 12. Configuración de la BD

Configuración del usuario que se pueda iniciar la sesión del BD

mysql -u oneadmin -p



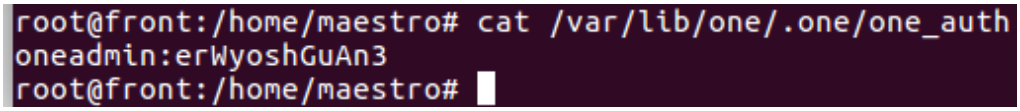
Captura 13. Ingreso al BD configurada

Configuración las credenciales de oneadmin

Anteriormente se genera un archivo aleatorio donde se encuentra ubicado en el archivo

/var/lib/one/.one/one_auth

cat /var/lib/one/.one/one_auth

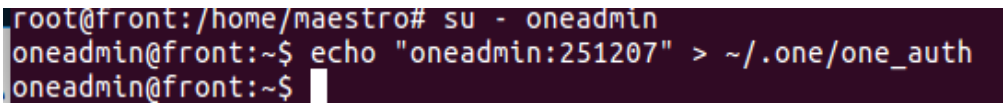


Captura 14. Clave por defecto del archivo one_auth

Si desea hacer el cambio de la clave se debe hacer aquí, donde se generará una clave de oneadmin en el primer arranque, el cual puede usar el presente comando para hacer combio en la contraseña onedamin

su - oneadmin

*echo "oneadmin:**251207**" > ~/.one/one_auth*



Captura 15. Cambio de clave en el archivo one_auth

Configure Firewall (UFW)

Si tiene un Firewall activo debe permitir los puertos específicos que es requerido para

Opnenebula,

Si se está ejecutando, se debe abrir el puerto requerido donde se accederá a la interfaz de Sunstone.

sudo ufw allow proto tcp from any to any port 9869

Start OpenNebula daemons.

Comando para iniciar los demonios de OpenNebula

systemctl start opennebula opennebula-sunstone

systemctl enable opennebula opennebula-sunstone

Verificación de la instalación y acceso de OpenNebula

Al iniciar por primera vez Opennebula se verifica que los comandos se conecten al demonio de OpenNebula.Para la verificación se puede hacer con CLI Linux o en la interfaz gráfica de usuario: Sunstone. Desde CLI en el front-end, ejecute el siguiente comando como oneadmin:

su - oneadmin -c "oneuser show"

```
root@front:/home/maestro# sudo ufw allow proto tcp from any to any port 9869
Rules updated
Rules updated (v6)
root@front:/home/maestro# systemctl start opennebula opennebula-sunstone
root@front:/home/maestro# systemctl enable opennebula opennebula-sunstone
Created symlink from /etc/systemd/system/multi-user.target.wants/opennebula.serv
ice to /lib/systemd/system/opennebula.service.
Created symlink from /etc/systemd/system/multi-user.target.wants/opennebula-suns
tone.service to /lib/systemd/system/opennebula-sunstone.service.
root@front:/home/maestro# su - oneadmin -c "oneuser show"
USER 0 INFORMATION
ID              : 0
NAME            : oneadmin
GROUP           : oneadmin
PASSWORD        : 0c13ca1d2481c79ee2b4914fc5c85fc19a8f6346
AUTH_DRIVER     : core
ENABLED         : Yes

TOKENS

USER TEMPLATE
TOKEN_PASSWORD="104dc490b9be14d21a8179938072960dfb4ba470"

VMS USAGE & QUOTAS

VMS USAGE & QUOTAS - RUNNING

DATASTORE USAGE & QUOTAS

NETWORK USAGE & QUOTAS

IMAGE USAGE & QUOTAS

root@front:/home/maestro#
```

Captura 16. Demonio activo de OPenNebula

Configuración Passwordless SSH

Se crea una clave para el oneadmin

```
logout
root@front:/home/maestro# passwd oneadmin
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

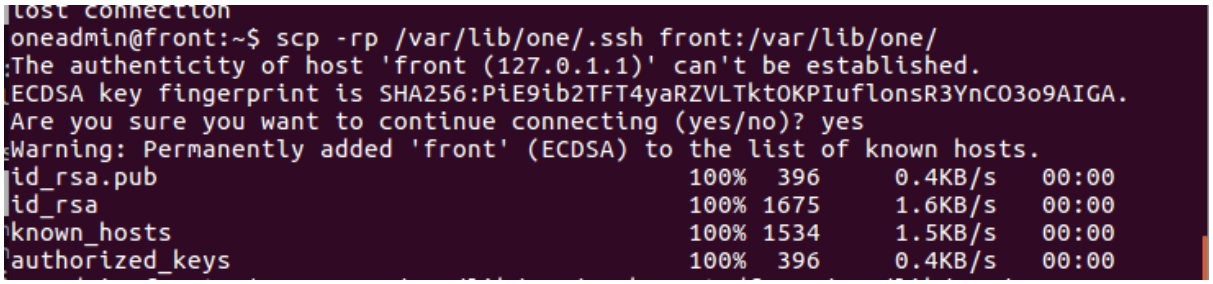
Captura 17. Creación de la clave oneadmin

```
root@front:/home/maestro# su - oneadmin
oneadmin@front:~$ ssh-keyscan front wantedface >> /var/lib/one/.ssh/known_hosts
# wantedface:22 SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8
# wantedface:22 SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8
# wantedface:22 SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8
oneadmin@front:~$
```

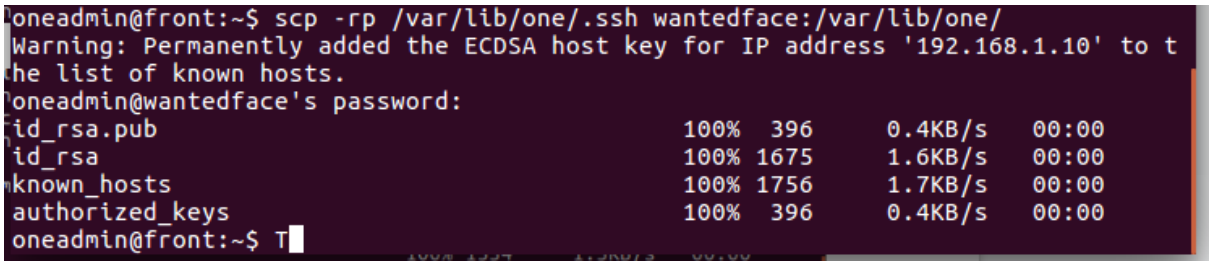
Captura 18. Ingreso al usuario oneadmin

Generación de las llaves

El levantamiento correcto de Opennebula y dar permisos requeridos al usuario oneadmin, lo primordial es la generación del respectivo archivo el contiene las llaves del Front como del nodo dentro de los directorios que presentan a continuación. En la presente imagen (Fig.,) genera las llaves tanto del Front como en nodo.



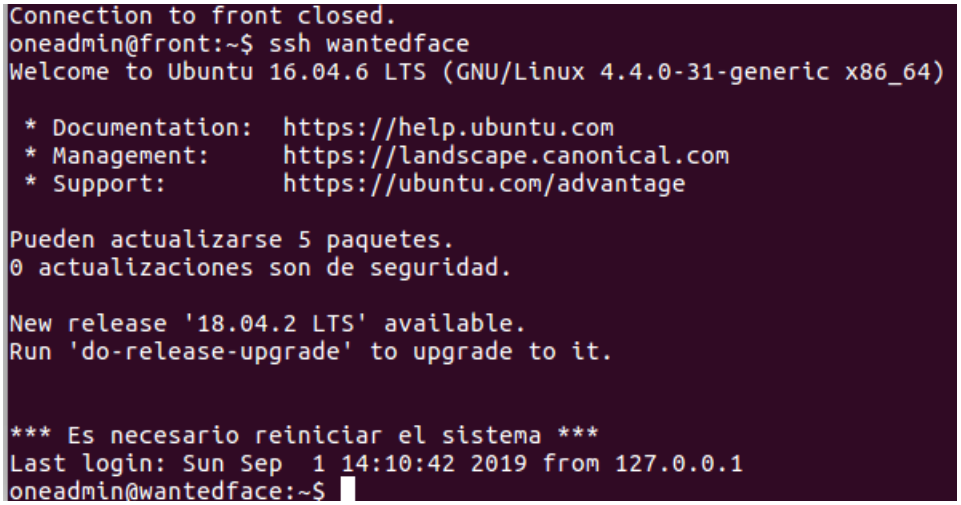
Captura 19. Comando que genera la llave en front



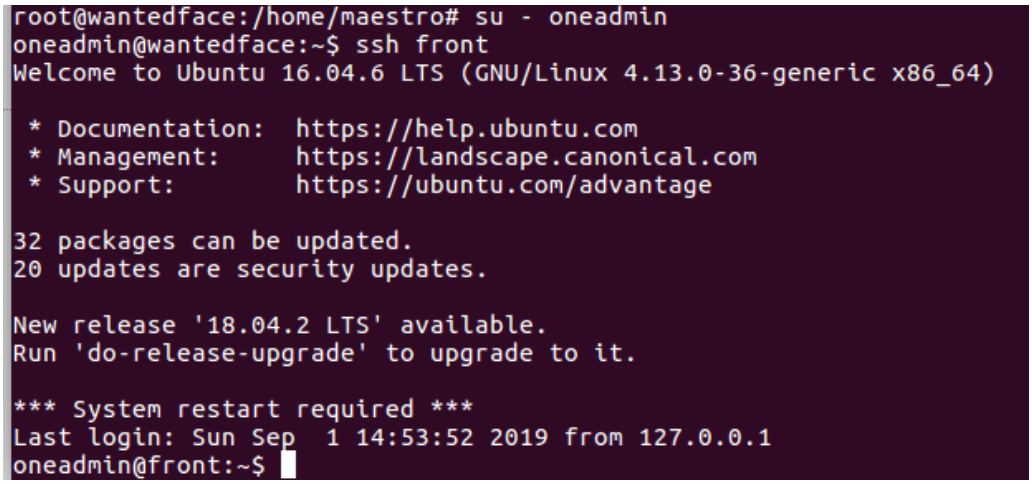
Captura 20. Comando que genera la llave en el nodo

Ingreso atravesó de SSh

Al crear las respectivas llaves y se han agregado al archivo, se realizan las pruebas de ingreso mediante ssh desde la parte del fron hacia al nodo y viceversa.



Captura 21. comando para agregar desde el front hacia al nodo

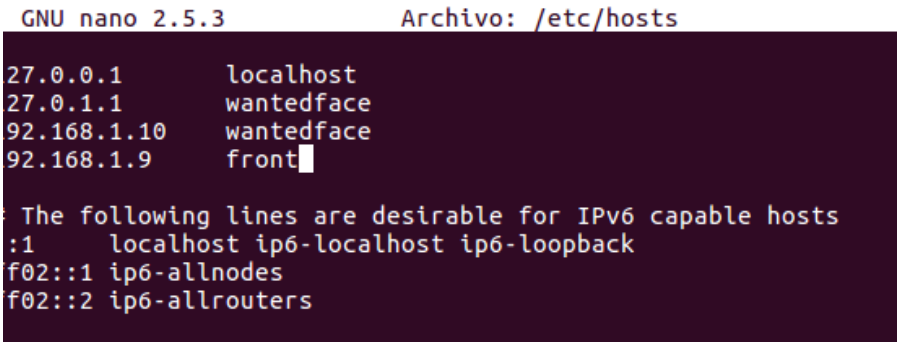


Captura 22. Comando para agregar desde el nodo hacia el front

INSTALACION OPENNEBULA KVM SOBRE UBUNTU 18.04

Para la configuración del nodo (KVM) configuración del hosts como se verá la presente imagen.

sudo /etc/hosts



Captura 23. configuración del hosts en nodo (KVM)

Configuración del bridge

Para la configuración del bridge en nodo se instala el paquete de bridge

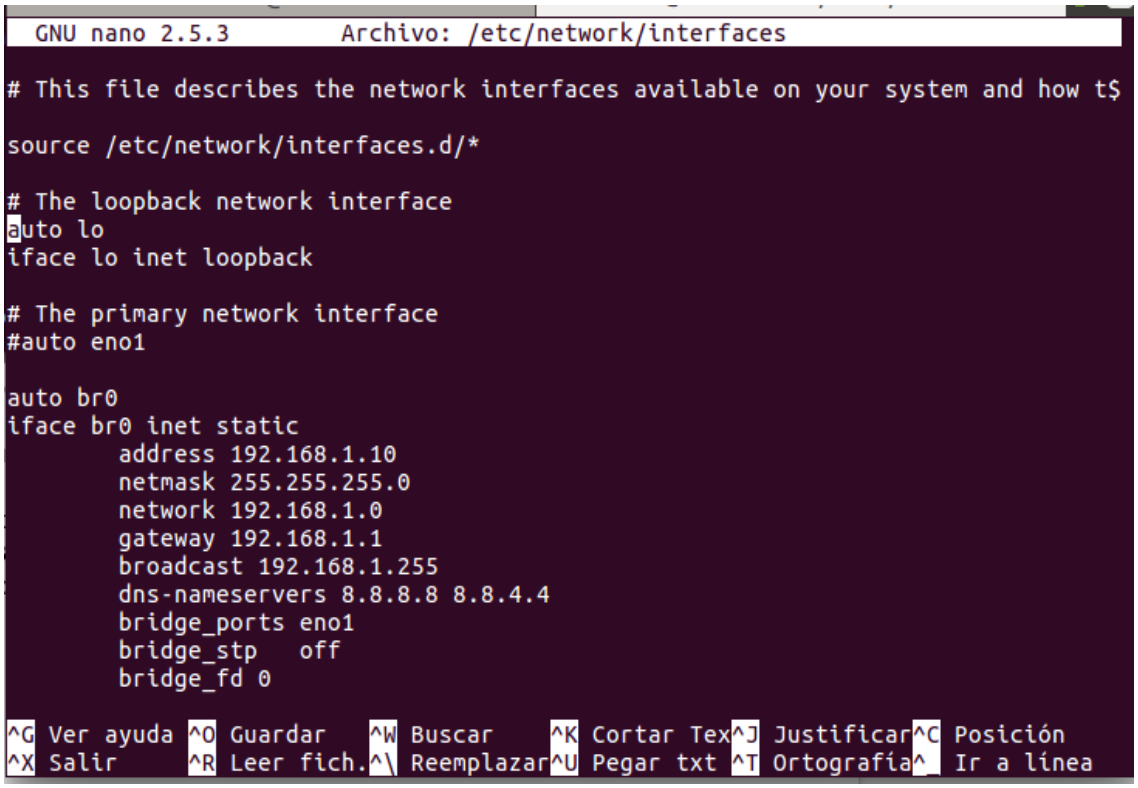
sudo apt-get install bridge-utils

Configuración del Bridge en nodo de opennebula

Dentro de la configuración de opennebula se ingresa mediante con siguiente

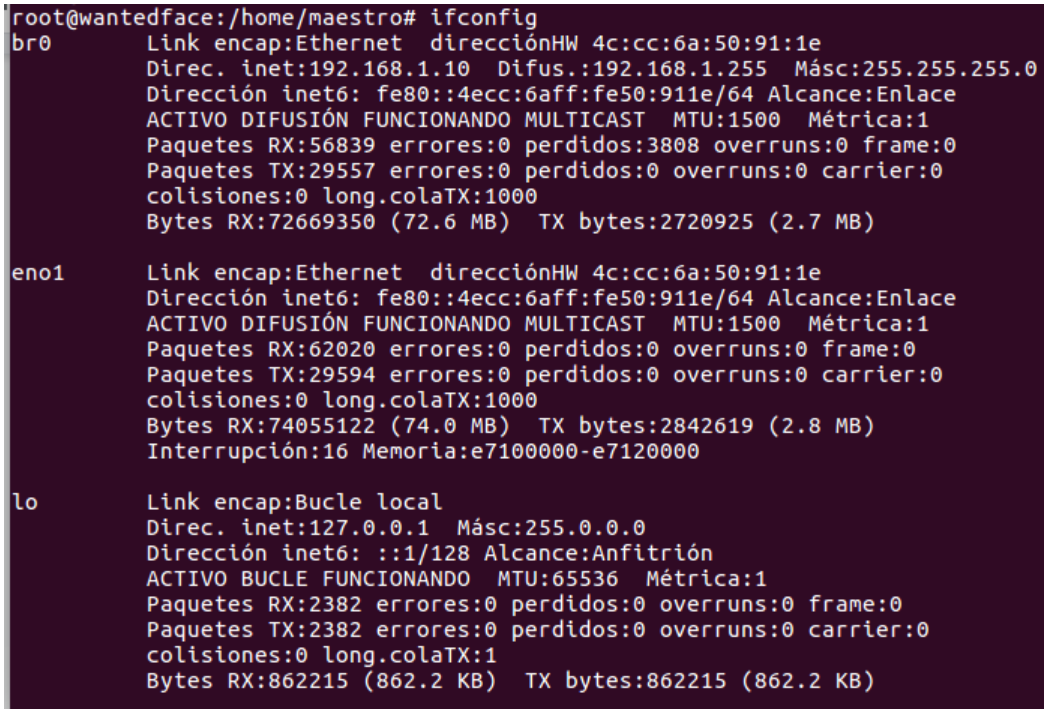
comando, y se procede la siguiente configuración del bridge. En la presente imagen (Fig., 24)

maestro@wantedface:~$ sudo nano /etc/network/interfaces



Captura 24. Configuración del bridge

Al terminar la configuración el bridge se manda a reiniciar el computador, con comando ifconfig se podrá ver si el bridge se actualizo. En la siguiente imagen (Fig.,25) se ve el bridge ya levantado

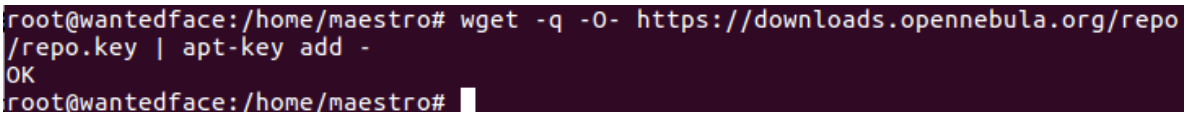


Captura 25. Bridge creado y levantado

Agregar el repositorio de OpenNebula.

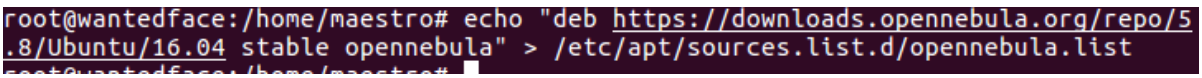
Agregación del repositorio para la instalación para Opennebula

wget -q -O- https://downloads.opennebula.org/repo/repo.key | apt-key add -



Captura 26. Agregación del repositorio

echo "deb https://downloads.opennebula.org/repo/5.8/Ubuntu/16.04 stable opennebula" > /etc/apt/sources.list.d/opennebula.list



Captura 27. Repositorio agregado al sistema

Al termino de agregar los repositorios se debe de actualizar todos los repositorios

sudo apt-get update

Instalacion del software del paque para el nodo (KVM), en OpenNebula

sudo apt-get install opennebula-node

sudo service libvirt-bin restart # ubuntu

Al terminar de la Instalación de todo el software de Opennebula se crea e

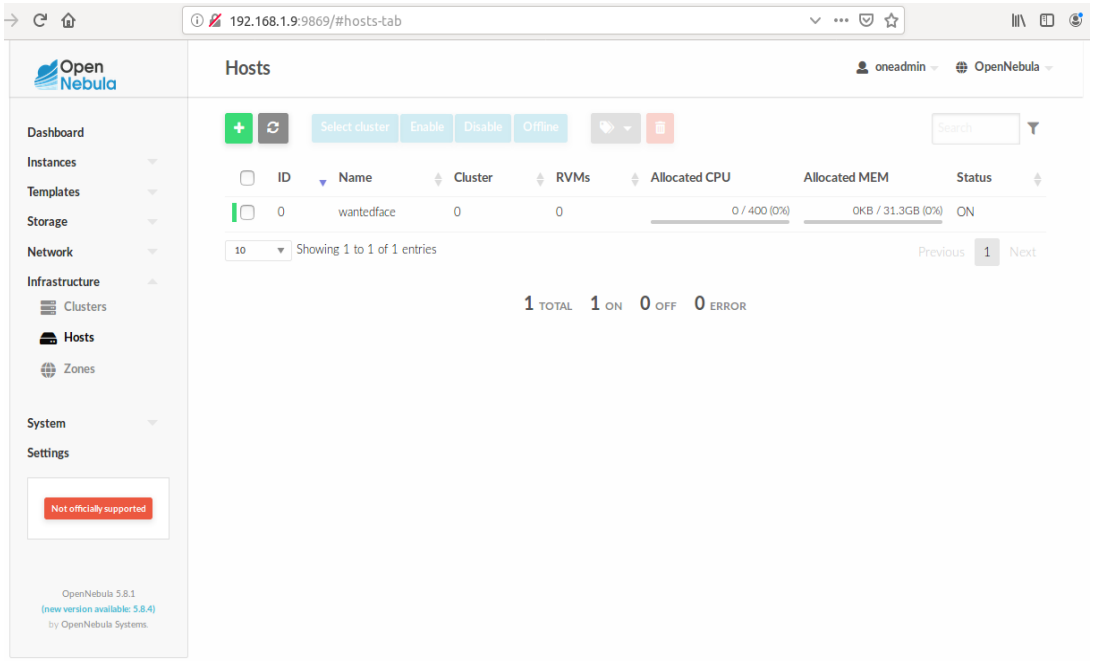
CREACIÓN HOST

Al finalizar la instalación y configuración del software de OpenNebula, se debe agregar un host a OpenNebula. Para iniciar máquinas virtuales en OpenNebula en el.

Agregar Host través de Sunstone

Dentro este se abre Sunstone, de allí nos dirigimos a **infraestructura** y damos en la opción **Host** y se da un cli en el botón y seleccionamos la opción **KVM**, configure la IP del nodo en el campo Nombre el host. Regrese a la sección de hosts y confirme que está en estado **ON**.

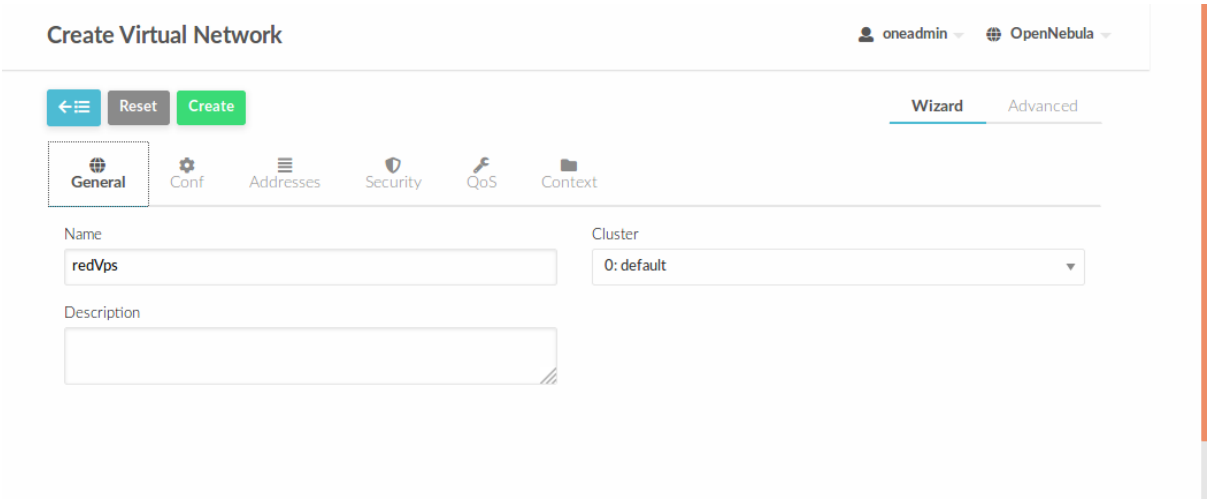
En la imagen presente se verá a continuación de la (Fig.,28) está en el activo



Captura 28. Activo del Host en ON

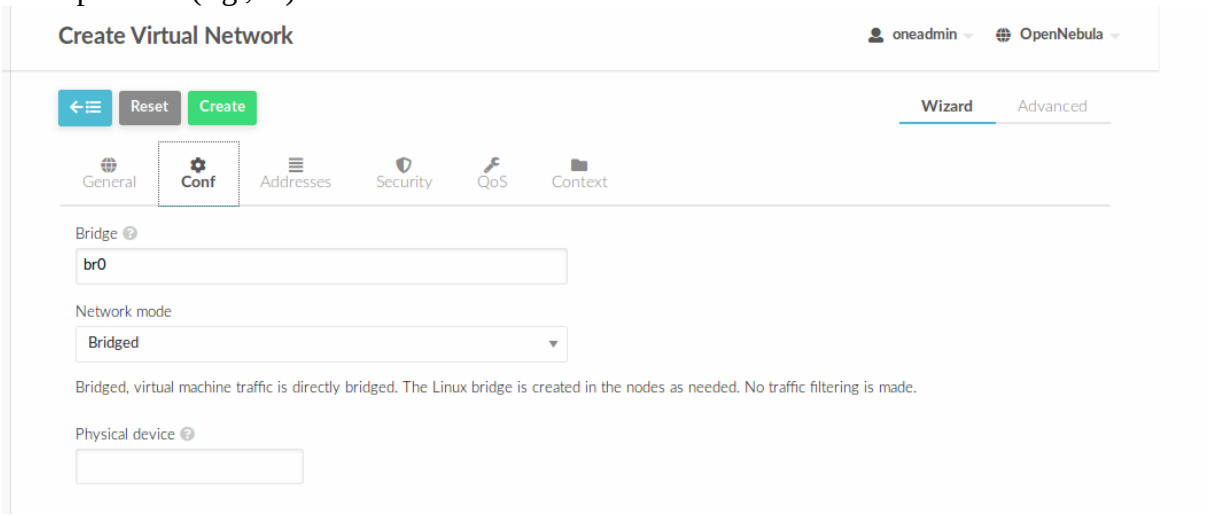
CONFIGURACIÓN DE RED

En el sub menú Nic0 de seleccionar la subred a la que queremos que pertenezca a la máquina virtual , ver que IPs disponibles

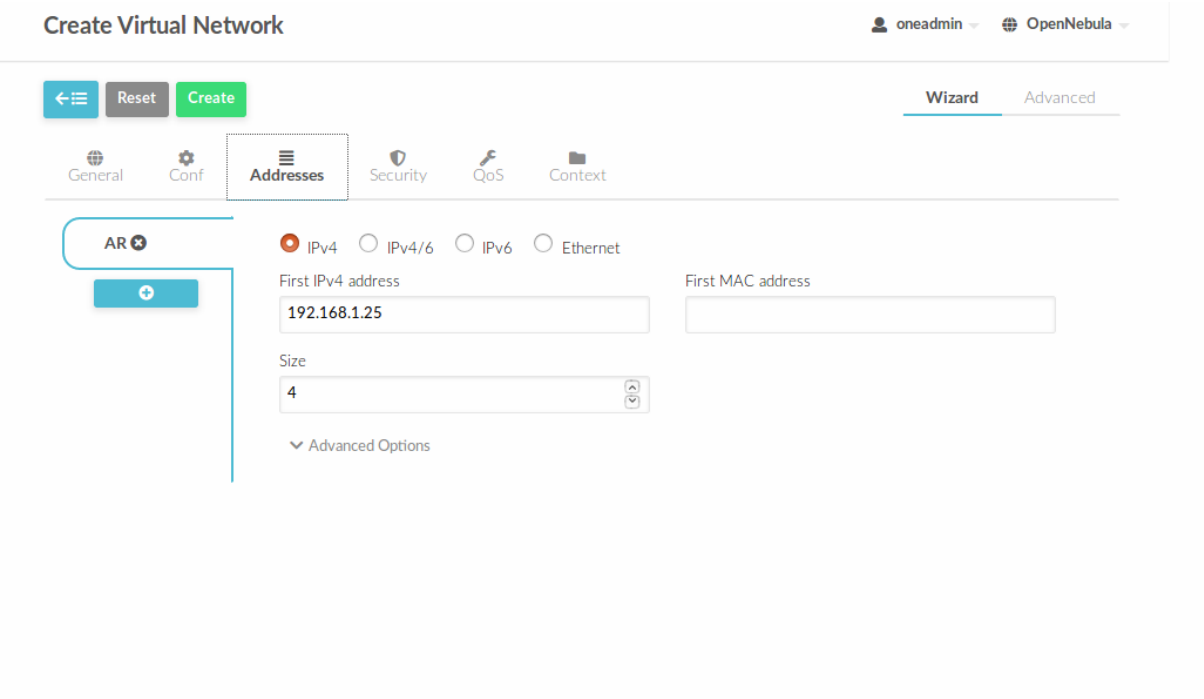


Captura 29. Creación de la Red

En la presente (fig.,30) se creará la



Captura 30. Creacion del br0 para la red



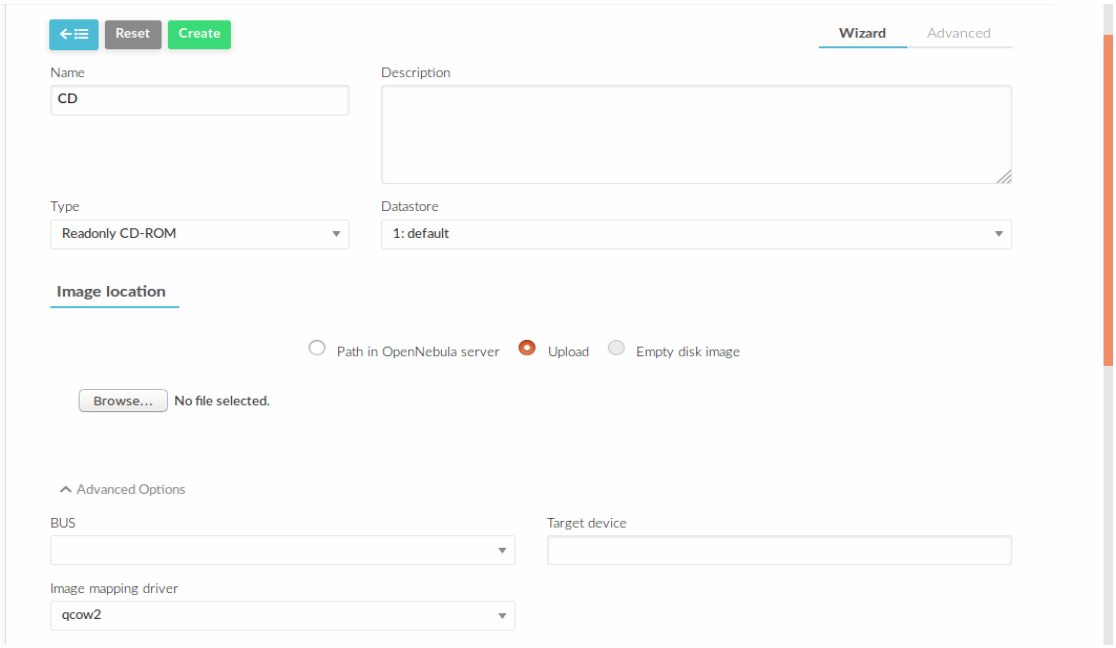
Captura 31. Creación de rango de IPs

IMAGES

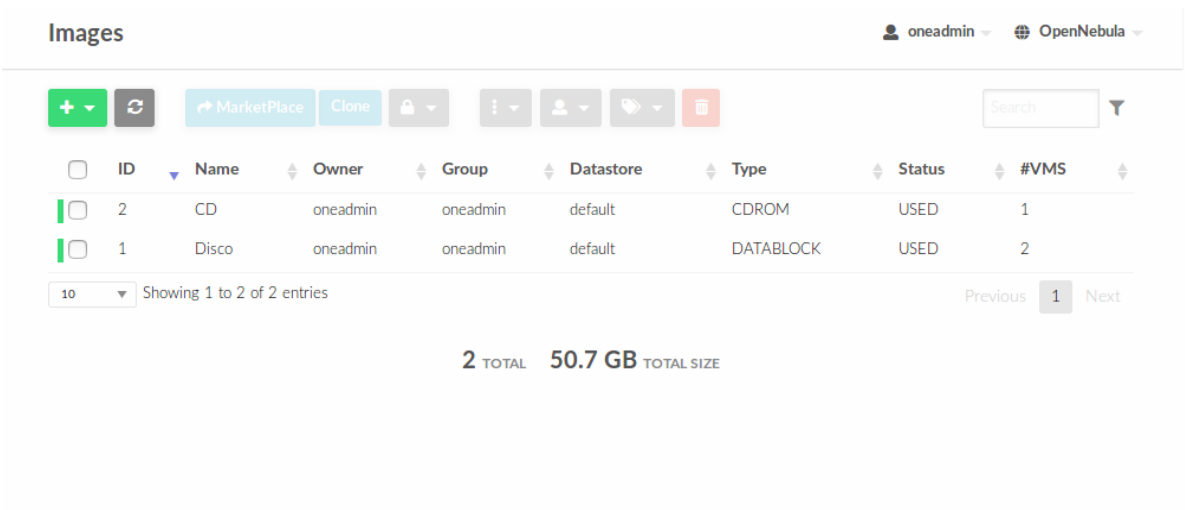
Ingresamos en el menú principal se debe dirigir al menú Store y Imagen, en donde se subirá la imagen del sistema Operativo que realizará y también se creará el disco duro para despliegue de las máquinas virtuales.

Para la subida la imagen del sistema operativo(iso) se debe de dar click en el icon de color verde, se abrirá una ventana en la que se creará la información siguiente

Name: Nombre del sistema opertivo(SO) que se utilizara
Tyoe: Readonly CD-ROM
Datastore: Defaul, Si se tiene más nodos seleccionar en donde desea guardar el archivo
Imagen Location: Upload – click en examinar y buscar el iso del sistema Operativo que se va utilizar.
Nota: En la opción Advanced Option, cambiar la imagen mapping driver por de raw a qcow2



Se da Click en crear y se debe de esperar hasta que se cree y suba la imagen



Para crear el disco duro virtual, seda en opción + de color verde se abrirá una ventana

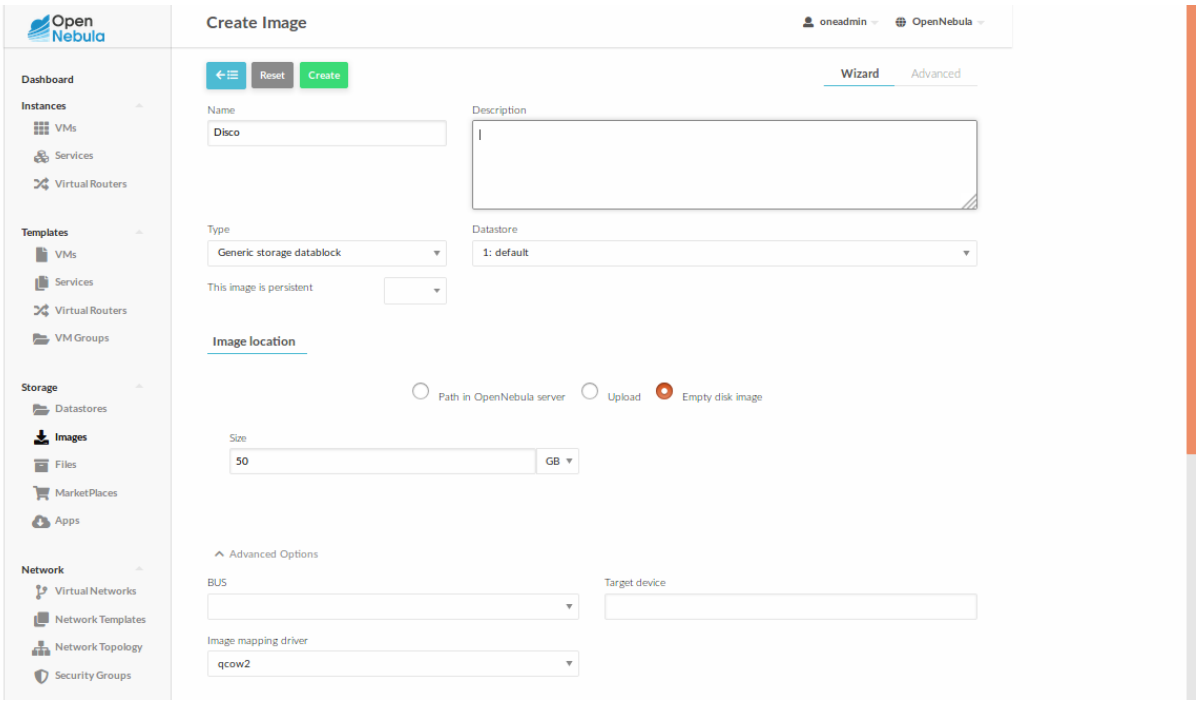
Name: Nombre del disco duro virtual que se usara

Tyoe: Generic Storage Datablock

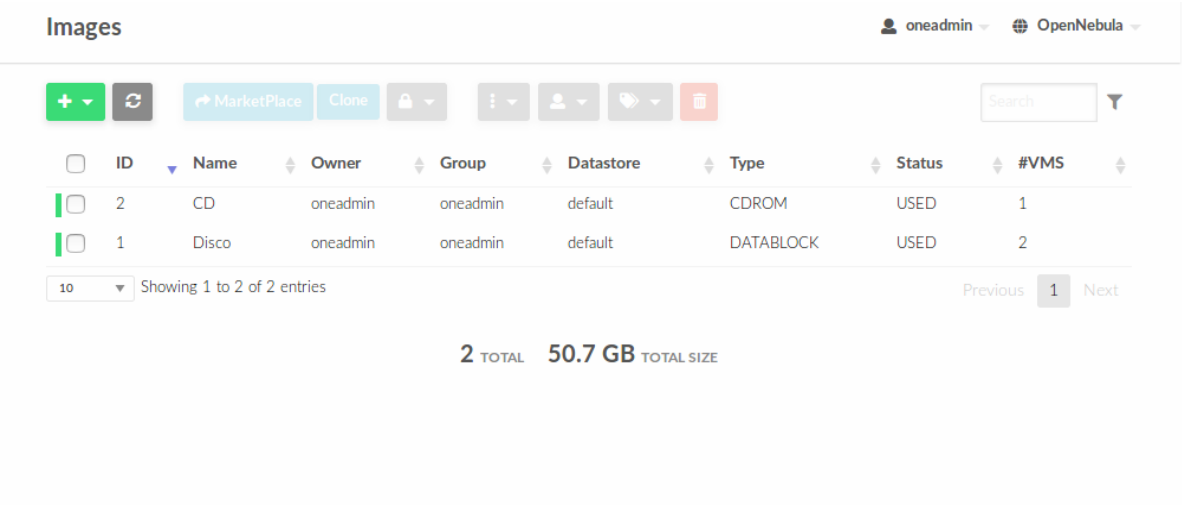
Datastore: Defaul, Si se tiene más nodos seleccionar en donde desea guardar el archivo

Imagen Location: Empty disk image – sede de poner el tamaño para el disco duro en Gb.

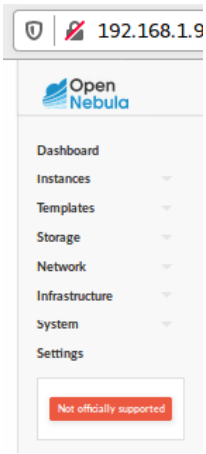
Nota: En la opción Advanced Option, cambiar la imagen mapping driver por de raw a qcow2



Dar un click en créate y esperar un tiempo mientras se crea el disco duro virtual.



Nos dirigimos al menú principal en la opción Templates - > VMs donde se creará la plantilla para desplegar máquinas virtuales



CREACION DEL TEMPLATE

Para la creación de la una plantilla se da click en créate donde se encontrará una nueva ventana con varios menús donde se definirá las características para la máquina virtual.

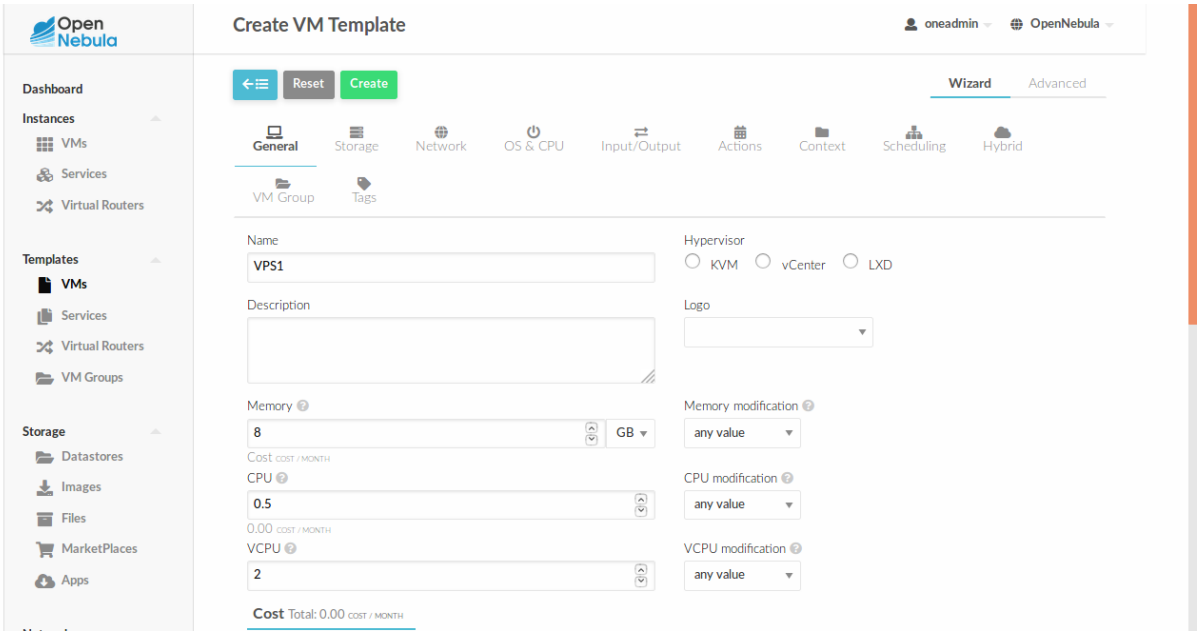
Menú General

Name: El nombre que se dara en la plantilla

Memory: Asignacion de la memoria RAM en GB

CPU: porcentaje del CPU físico para 100, ejemplo 0.5

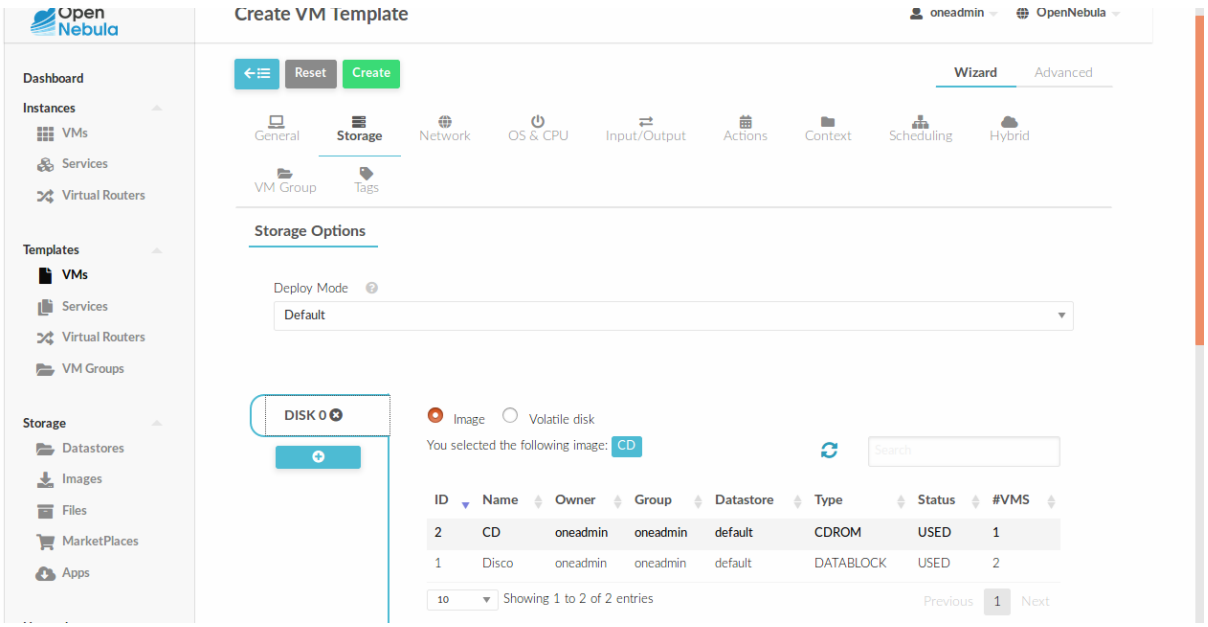
VCPU: Este es opcional, este indica un procesador virtual para la máquina virtual

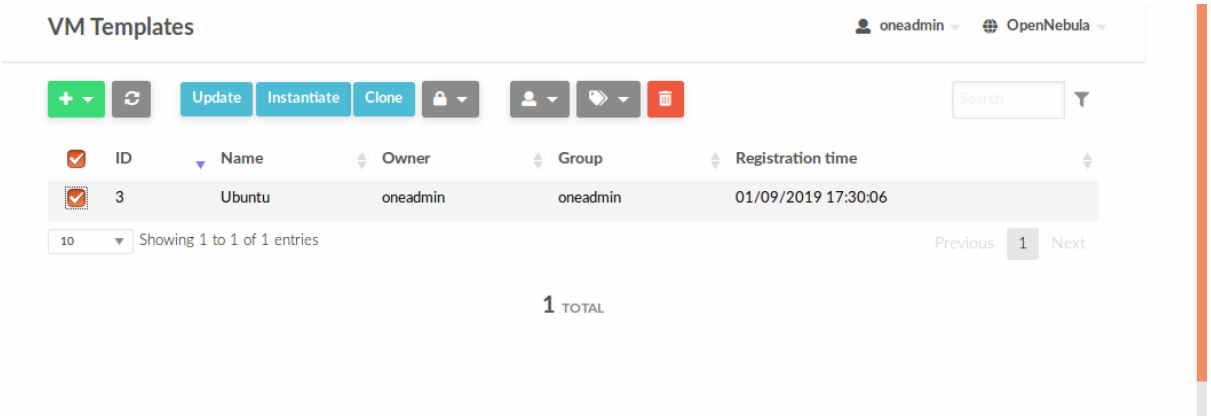


Menú Storage

En el submenú Disko 0 se elige la imagen del sistema operativo que se creó anteriormente.

En el submenú Disko 1 se elige el disco duro virtual que se creó anteriormente.





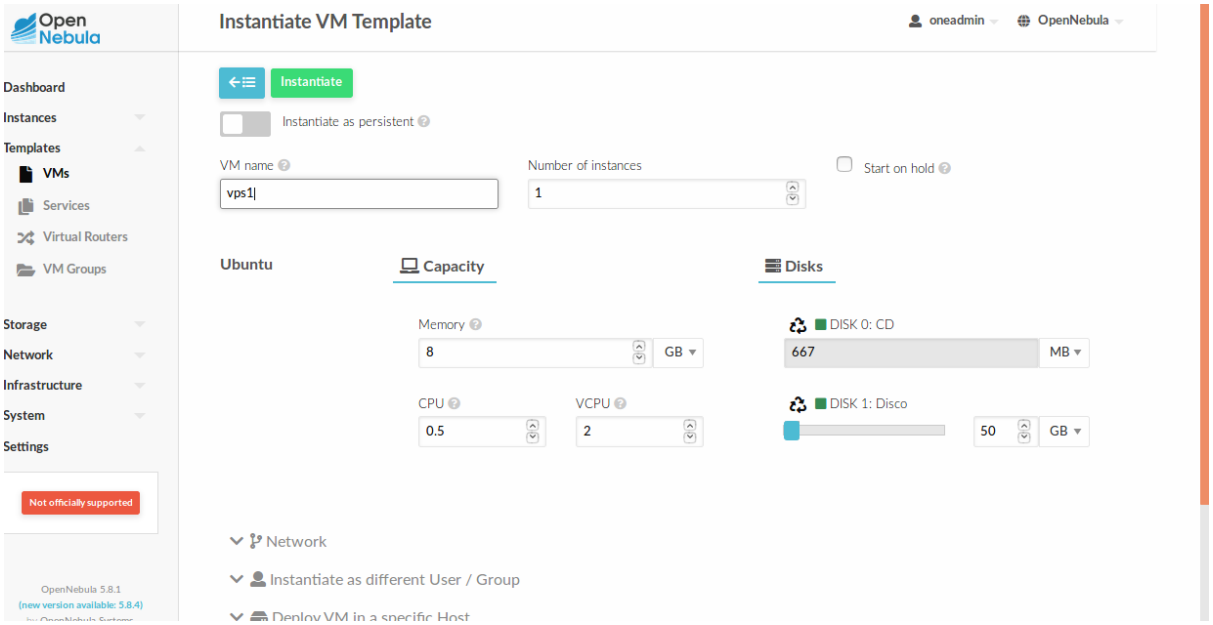
En el menú Templates -> VMs se da un click en la palntilla que se creo

Para instanciar el vps se abre una ventana de la definición de la plantilla para máquina virtual.

Damos click en el icono azul Instance.

Se da nombre a la máquina virtual en este caso es vps1

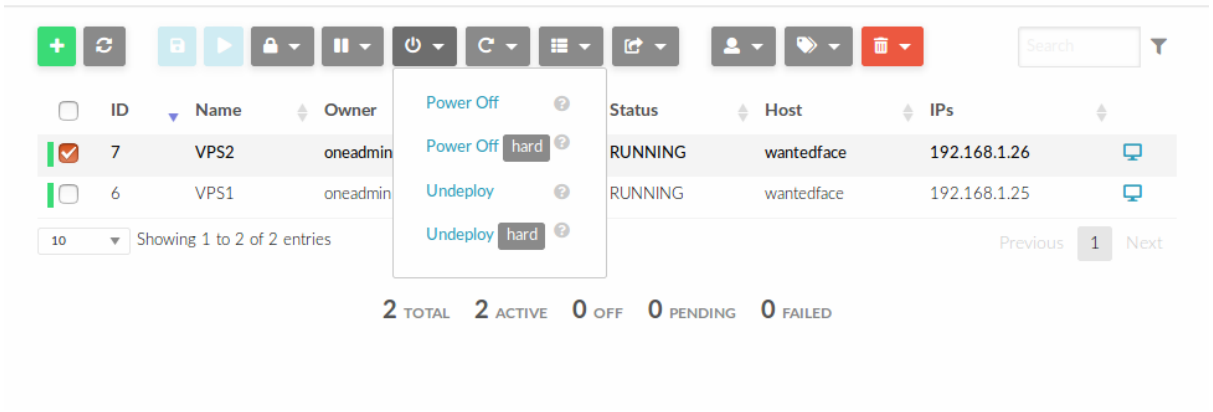
Nota: Nos dirigimos Desplegar Advance Options y se selecciona e Host y el DataStore adecuado para el despliegue de la maquina virtual.

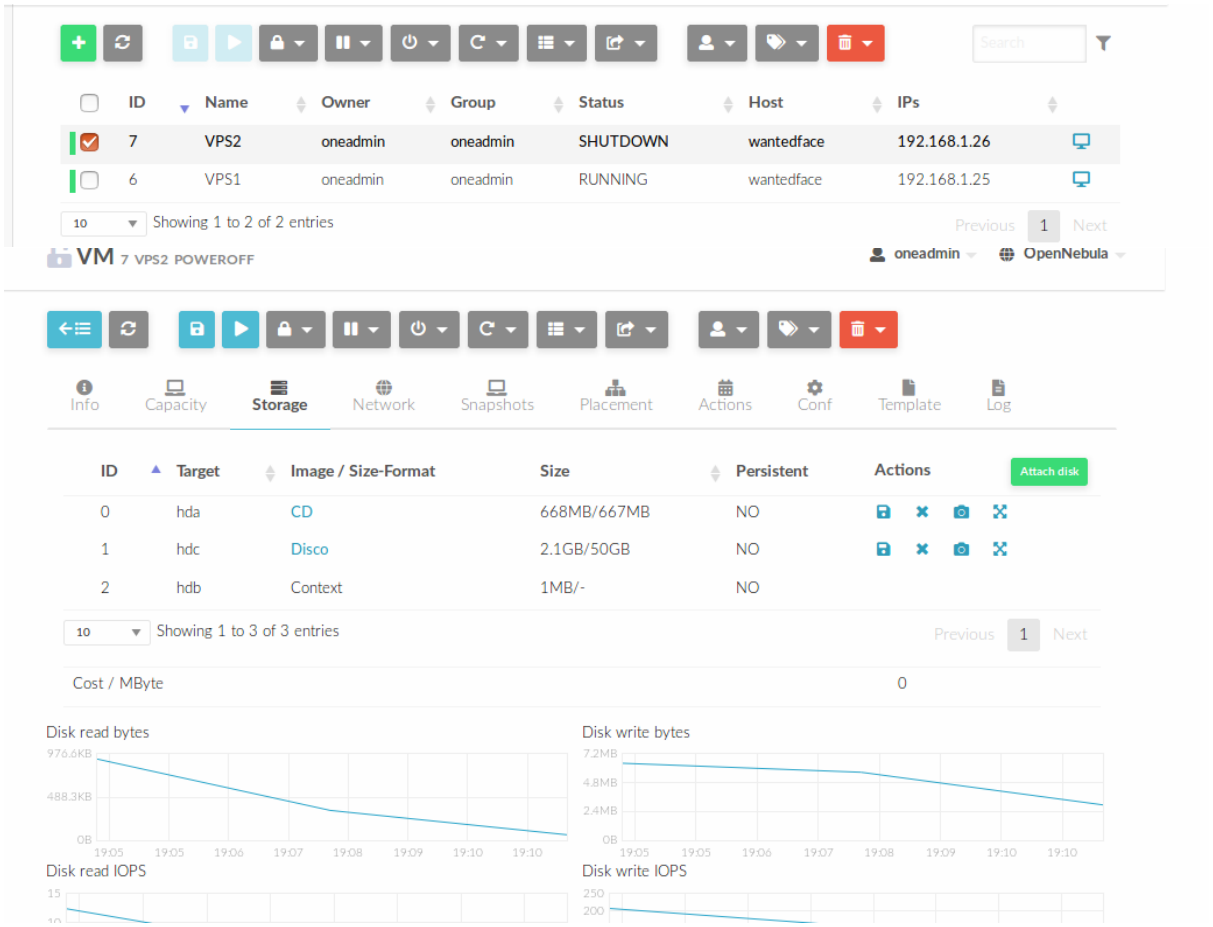


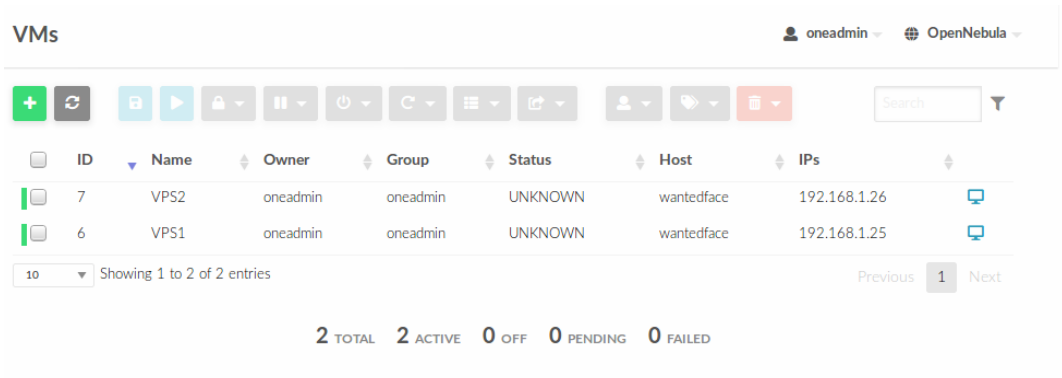
TERMINADO DE LA INSTALACION DEL SISTEMA OPERATIVO

Damos click en el icono verde de Instance, esperamos el mensaje de confinación de parte de creación inferior, derecha de la ventana.

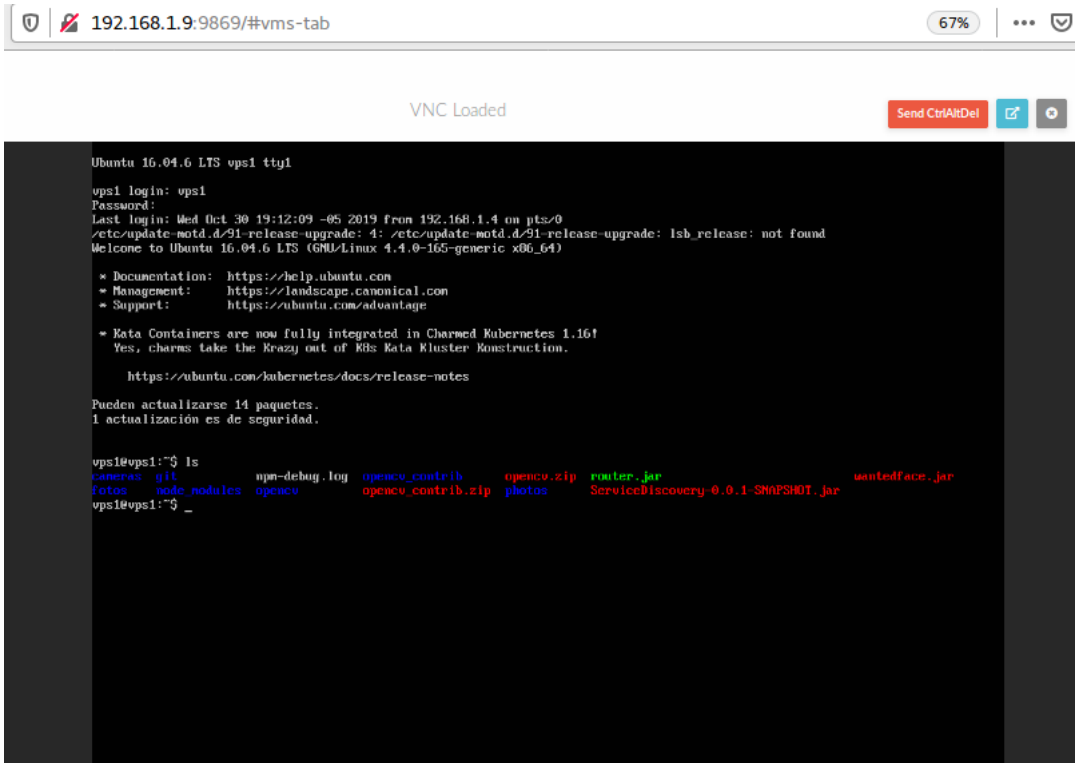
Dentro del menú de Instance => VMs se tiene máquinas virtuales que se creó, el Status PENDING es un proceso en cual se están copiando los archivos adecuados para el despliegue, hacen comprobaciones de la existencia de los recursos subyacentes para su funcionamiento. Se observará el despliegue que fue exitoso ya que se puede usar la máquina virtual cuando status diga RUNNING.







Como se puede ver, en donde se realizará la utilización de la máquina virtual.



Anexo 3: Docker.

DOCKER

Para el proceso de Docker se necesita lo siguientes.

- Crear una cuenta en Docker Hub, si desea la creación de las imágenes y hacer push al Docker Hub.

1. Instalación Docker

   Se instalará docker desde un repositorio oficial de docker para poder obtener la última versión

   Actualización los repositorios

   ```
   sudo apt update
   ```

   se debe agregar las claves al servidor como se procederá la siguiente línea de comandos.

   sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D

   sudo apt-add-repository 'deb https://apt.dockerproject.org/repo ubuntu-xenial main'
   sudo apt-get update

apt-cache policy docker-engine
sudo apt-get install -y docker-engine

2. Estado del Docker

   Al terminar de la instalación de Docker procedemos utilizar el siguiente comando para poder verificar la instalación de Docker.

```
sudo systemctl status docker

Processing triggers for ureadahead (0.100.0-19) ...
root@vps1:/home/vps1# sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: e
   Active: active (running) since dom 2019-10-20 12:17:17 -05; 3min 12s ago
         Docs: https://docs.docker.com
 Main PID: 6991 (dockerd)
   CGroup: /system.slice/docker.service
         ├─6991 /usr/bin/dockerd -H fd://
         └─7010 docker-containerd -l unix:///var/run/docker/libcontainerd/dock

oct 20 12:17:16 vps1 dockerd[6991]: time="2019-10-20T12:17:16.268039344-05:00" l
oct 20 12:17:16 vps1 dockerd[6991]: time="2019-10-20T12:17:16.268369382-05:00" l
oct 20 12:17:16 vps1 dockerd[6991]: time="2019-10-20T12:17:16.268632232-05:00" l
oct 20 12:17:16 vps1 dockerd[6991]: time="2019-10-20T12:17:16.269446624-05:00" l
oct 20 12:17:16 vps1 dockerd[6991]: time="2019-10-20T12:17:16.651253416-05:00" l
oct 20 12:17:17 vps1 dockerd[6991]: time="2019-10-20T12:17:17.015109368-05:00" l
oct 20 12:17:17 vps1 dockerd[6991]: time="2019-10-20T12:17:17.875727908-05:00" l
oct 20 12:17:17 vps1 dockerd[6991]: time="2019-10-20T12:17:17.876170528-05:00" l
oct 20 12:17:17 vps1 systemd[1]: Started Docker Application Container Engine.
oct 20 12:17:17 vps1 dockerd[6991]: time="2019-10-20T12:17:17.898174805-05:00" l
lines 1-19/19 (END)
```

3. Descargando el Docker del Repositorio

   root@vps1:/home/vps1# clear

   root@vps1:/home/vps1# docker pull anitas/imagenjava-subido
   Correr el docker
   root@vps2:/home/vps2# docker pull anitas/imagenjava-subido

4. Ejecución del Docker

   Mandamos a ejecutar el docker como se ver a continuación:
   wantedfaceD => como voy a llamar el docker
   -it => para que sea interactivo
   anitas/wantedface-subido => el nombre de mi imagen
   exit => para salir

Ejemplo:

docker run --name wantedfaceD -it anitas/wantedface-subido

5. Crear red en el docker

Comando para creación de nuestra propia redes

docker network create test1
desconectar el
docker network disconnect bridge wntedfaceA

CONECCTAR MI CONTENEDOR AL CONTENEDOR QUE CREO

docker network connect test wntedfaceA

docker network inspect test

CONFIGURACION

Configuración del docker para la red

Comando para ver las redes Como puede ser:

- ✓ docker network ls
- ✓ birdge =>
- ✓ host => red por defecto que el contenedor se vincula de la red bride
- ✓ docker network inspect bridge => para ver lo que contiene dentro de la red
- ✓ docker network inspect bridge => Permite ver

Creación de nuestra propia red

Esta genera una red para un grupo de contenedores para la creación de una red nueva con un rango de ip, como se podrá ver la siguiente línea de comandos.

docker network create --subnet 10.10.1.0724 --gateway 10.10.1.254 dmz
(dmz=>nombre de red)
Para verificar el listado del docker.
docker network ls

Limpiar memoria control f5 de la web, se crear un contenedor que utilizara esta red en paticular dz como a continuacion de vera las siguientes líneas de comando:

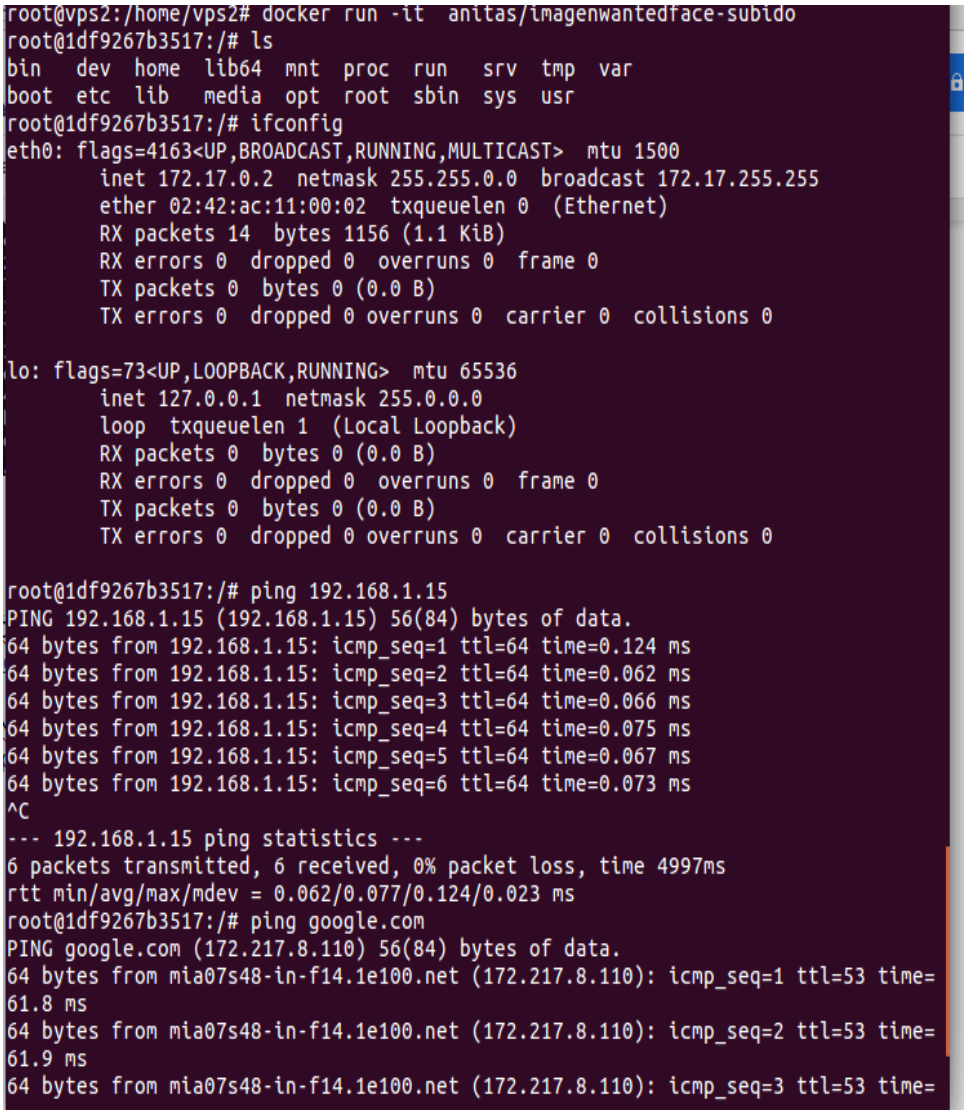
- ✓ -itd => interactivo de ta
- ✓ --name => dmz nombre del contenedor

- ✓ - h => dmz nombre del host
- ✓ --network => dmz nombre de la red que se crea

Ejemplo: para la ejecucion.

docker run - itd --name dmz -h dmz --network dmz debian

En la siguiente imagen se puede apreciar, el funcionamiento de red dentro del contenedor.



6. Instalación de paquetes en Docker.

Para poder instalar paquetes dentro docker se debe, estar dentro de ello y actualizar si no se actualiza no se podrá hacer nada.

- ✓ root@7f7f1a60d953:/# apt-get update
- ✓ root@7f7f1a60d953:/# apt-get install python3-dev python3-pip

- ✓ root@7f7f1a60d953:/# sudo apt install python-pip
- ✓ root@7f7f1a60d953:/# pip3 install flask
- ✓ root@7f7f1a60d953:/# pip install Flask
- ✓ root@7f7f1a60d953:/# apt install maven
- ✓ root@7f7f1a60d953:/# mkdir git
- ✓ root@7f7f1a60d953:/# pip install websocket
- ✓ root@7f7f1a60d953:/# pip3 install Flask
- ✓ root@7f7f1a60d953:/# pip install request pip install websocket-client

7. Para hacer un Contenedor sea Interactivo.

Este permitirá ser interactivo el contendor, una vez terminado de ejecutar el comando attach ID del contenedor.

```
root@vps1:/home/vps1# docker attach 7f7f1a60d953
root@vps1:/home/vps1# docker attach 7f7f1a60d953
root@7f7f1a60d953:/# ls
bin  boot  dev  etc  git  home  lib  lib64  media  mnt  opt  proc  root  run
sbin  srv  sys  tmp  usr  var
```

En este comando podemos ver los contenedores

```
root@7f7f1a60d953:/# docker ps -a
```

8. Hacemos un commint de Contenedor

Hacemos un commit permitiendo guardar los cambios y como se podrá ver:

```
root@vps1:/home/vps1# docker commit 7f7f1a60d953 anitas/imagenjava-tesis
sha256:287cd538f36b840f8ff564fff2d8f4d07956c54b0d0b9ce07b3a83a7376147d1
```

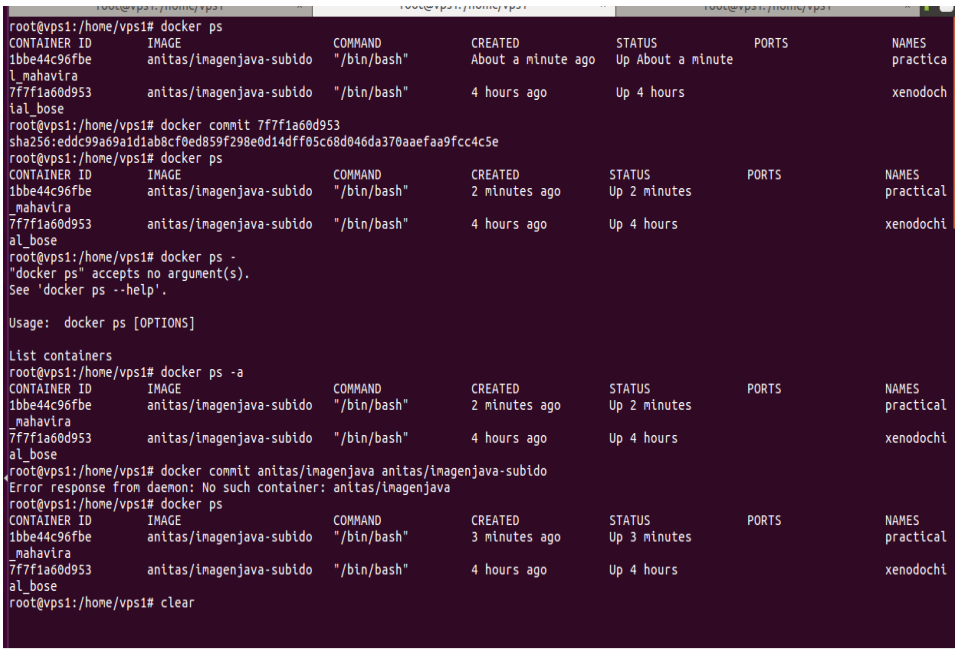
Para ver el ID del contenedor el nombre del contenedor.

```
root@vps1:/home/vps1# docker ps
CONTAINER ID   IMAGE                          COMMAND       CREATED
   STATUS          PORTS          NAMES
1bbe44c96fbe   anitas/imagenjava-subido   "/bin/bash"   5 minutes ago
   Up 5 minutes                   practical_mahavira
7f7f1a60d953   anitas/imagenjava-subido   "/bin/bash"   4 hours ago   Up 4
hours                          xenodochial_bose
```

Con el comando docker images permite ver o verificar de la imagen si se creó o no, se podrá ver a continuación, el TAG, ID,etc.

root@vps1:/home/vps1# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
anitas/imagenjava-tesis latest 287cd538f36b 2 minutes ago 2.28GB

En presente Imagen se podrá observar la interacción de los diferentes procesos se realizo anteriormente.

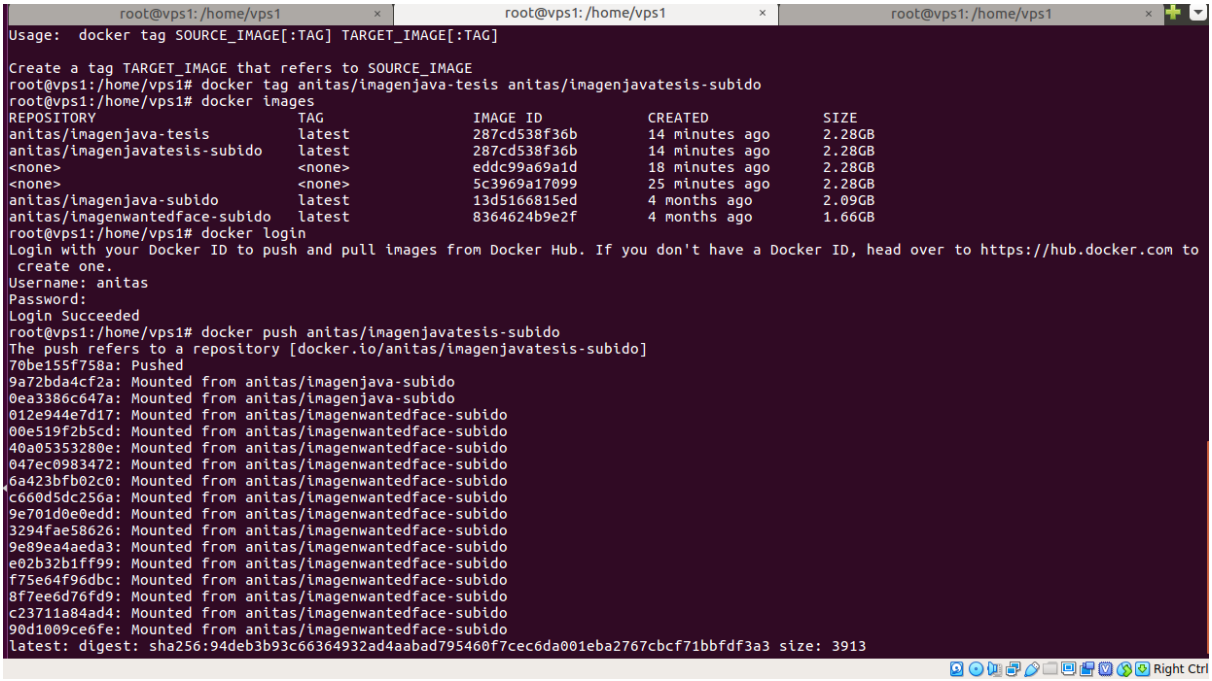


9. Mandamos a correr el contenedor

root@vps1:/home/vps1# docker run --name imagenjavatesis -d anitas/imagenjava-tesis
4b428340eac3d535cbe868c363faaf6c8147939067a34c5777c6d40455a3ff1e
root@vps1:/home/vps1#

10. Comando Tag

root@vps1:/home/vps1# docker tag anitas/imagenjava-tesis anitas/imagenjavatesis-subido
root@vps1:/home/vps1# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
anitas/imagenjava-tesis latest 287cd538f36b 14 minutes ago 2.28GB
anitas/imagenjavatesis-subido latest 287cd538f36b 14 minutes ago 2.28GB
<none> <none> eddc99a69a1d 18 minutes ago 2.28GB
<none> <none> 5c3969a17099 25 minutes ago 2.28GB
anitas/imagenjava-subido latest 13d5166815ed 4 months ago 2.09GB
anitas/imagenwantedface-subido latest 8364624b9e2f 4 months ago 1.66GB

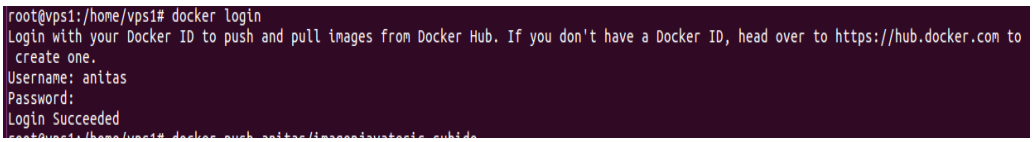


11. Logueo al Hub

Se debe de crear un repositorio en hub para esto se crea una cuenta

una vez ya creada, procedemos ingresar con las credenciales.

usuario docker: anitas
pass: Anita1234
Como se puede ingresar las siguientes credenciales para el logueo acceder a la cuenta y poder subir al repositorio.



12. Subir al repositorio hub

Proceso de despliegue a la web, se tarda un tiempo hasta subir al repositorio,

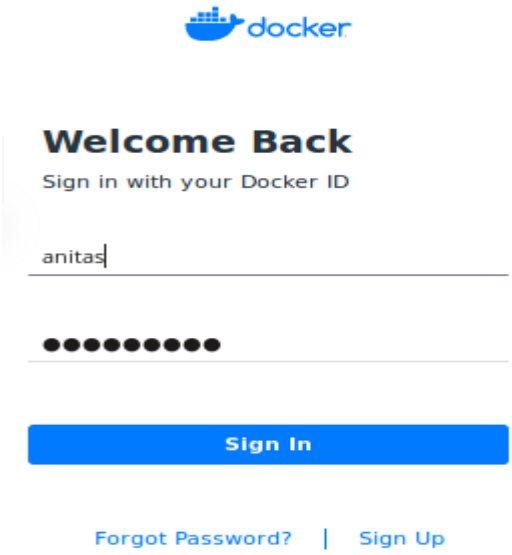
esperamos.

root@vps1:/home/vps1# docker push anitas/imagenjavatesis-subido

```
root@vps1:/home/vps1# docker push anitas/imagenjavatesis-subido
The push refers to a repository [docker.io/anitas/imagenjavatesis-subido]
70be155f758a: Pushed
9a72bda4cf2a: Mounted from anitas/imagenjava-subido
0ea3386c647a: Mounted from anitas/imagenjava-subido
012e944e7d17: Mounted from anitas/imagenwantedface-subido
00e519f2b5cd: Mounted from anitas/imagenwantedface-subido
40a05353280e: Mounted from anitas/imagenwantedface-subido
047ec0983472: Mounted from anitas/imagenwantedface-subido
6a423bfb02c0: Mounted from anitas/imagenwantedface-subido
c660d5dc256a: Mounted from anitas/imagenwantedface-subido
9e701d0e0edd: Mounted from anitas/imagenwantedface-subido
3294fae58626: Mounted from anitas/imagenwantedface-subido
9e89ea4aeda3: Mounted from anitas/imagenwantedface-subido
e02b32b1ff99: Mounted from anitas/imagenwantedface-subido
f75e64f96dbc: Mounted from anitas/imagenwantedface-subido
8f7ee6d76fd9: Mounted from anitas/imagenwantedface-subido
c23711a84ad4: Mounted from anitas/imagenwantedface-subido
90d1009ce6fe: Mounted from anitas/imagenwantedface-subido
latest: digest: sha256:94deb3b93c66364932ad4aabad795460f7cec6da001eba2767cbcf71bbfdf3a3 size: 3913
```

13. Ingreso sesión

Ingreso a la sesión para ver si subió el docker al repositorio.



14. Contendor en el repositorio web

Como se podrá observar subió exitosamente al repositorio de docker hub.