UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CARRERA DE TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR

Previo a la obtención del título de:

INGENIERO EN TELECOMUNICACIONES

Implementación de un sistema de seguridad interna basado en reconocimiento facial con plataforma de servicio inteligente

AUTOR:

Dominguez Silvestre Carmen Angelina

DOCENTE TUTOR:

Ing. Manuel Montaño Blacio. MSc.

LA LIBERTAD - ECUADOR

Año 2025

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Rovira Jurado, PhD.
DIRECTOR DE LA CARRERA

Ing. Manuel Montaño Blacio, MSc.
DOCENTE TUTOR

Ing. Daniel Jaramillo Chamba, Mgtr.
DOCENTE ESPECIALISTA

Ing. Luis Amaya Fariño, Mgtr .
DOCENTE GUÍA UIC

Ing. Corina Gonzabay De La A, Mgtr.
SECRETARIA

APROBACIÓN DE DOCENTE TUTOR

En mi calidad de docente tutor del trabajo de Integración Curricular denominado: **"Implementación de un sistema de seguridad interna basado en reconocimiento facial con plataforma de servicio inteligente"**, elaborado por **Carmen Angelina Dominguez Silvestre** estudiantes de la Carrera de Telecomunicaciones, Facultad de Sistemas y Telecomunicaciones de la Universidad Estatal Península de Santa Elena, previo a la obtención del título de Ingeniería en Telecomunicaciones, me permito declara que, tras supervisar el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos. En consecuencia, lo considero apto en todos sus aspectos y listo para ser evaluado por el docente especialista

Atentamente,

Ing. Manuel Montaño Blacio. MSc.

DOCENTE TUTOR

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA

DE SANTA ELENA

FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CERTIFICACIÓN

Certifico que luego de haber dirigido científica y técnicamente el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos, razón por el cual apruebo en todas sus partes el presente trabajo de titulación que fue realizado en su totalidad por **Carmen Angelina Dominguez Silvestre**, como requerimiento para la obtención del título de Ingeniero en Telecomunicaciones.

La Libertad, a los 30 días del mes de junio del año 2025

TUTOR

Ing. Manuel Montaño Blacio, MSc.

APROBACIÓN DE DOCENTE ESPECIALISTA

En mi calidad de docente especialista del trabajo de Integración Curricular denominado: **"Implementación de un sistema de seguridad interna basado en reconocimiento facial con plataforma de servicio inteligente"**, elaborado por **Carmen Angelina Dominguez Silvestre** estudiantes de la Carrera de Telecomunicaciones, Facultad de Sistemas y Telecomunicaciones de la Universidad Estatal Península de Santa Elena, previo a la obtención del título de Ingeniería en Telecomunicaciones, me permite declarar que, tras supervisar el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos. En consecuencia, lo considero apto en todos sus aspectos y listo para la sustentación del trabajo.

Atentamente,

DANIEL ARMANDO JARAMILLO CHAMBA

Firmado digitalmente por DANIEL ARMANDO JARAMILLO CHAMBA

Ing. Daniel Jaramillo Chamba. Mgtr.

DOCENTE ESPECIALISTA

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

AUTORIZACIÓN

Yo, **Carmen Angelina Dominguez Silvestre**

Autorizo a la Universidad Estatal Península de Santa Elena, para que haga de este trabajo de titulación parte de él, un documento disponible para su lectura consulta y procesos de investigación, según las normas de la Institución.

Cedo los derechos en línea patrimoniales de artículo profesional de alto nivel con fines de difusión pública, además apruebo la reproducción de este artículo académico dentro de las regulaciones de la Universidad, siempre y cuando esta reproducción no suponga una ganancia económica y se realice respetando mis derechos de autor.

La Libertad, a los 30 días del mes de Junio del año 2025

Carmen Angelina Dominguez Silvestre

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
DECLARACIÓN DE RESPONSABILIDAD

Yo, Carmen Angelina Dominguez Silvestre

DECLARAMOS QUE:

El trabajo de Titulación, **Implementación de un sistema de seguridad interna basado en reconocimiento facial con plataforma de servicio inteligente**, previo a la obtención del título en Ingeniero en Telecomunicaciones, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de nuestra total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

La Libertad, a los 30 días del mes de Junio del año 2025

AUTOR

Carmen Angelina Dominguez Silvestre

AGRADECIMIENTO

Agradecer a Dios por brindarme sabiduría, fortaleza y salud necesaria para terminar esta etapa universitaria. Al no soltarme de inicio a fin en situaciones complicados, en cada desafío que se presentaba en el día a día y en cada logro también, por eso gracias eternamente porque si no es por su amor y su misericordia nada habría sido posible.

A mis padres, Alcides y Carmen, gracias a ellos profundamente por ser ese pilar fundamental en toda mi vida estudiantil, por enseñarme el valor de la perseverancia de seguir adelante a pesar de toda situación y el esfuerzo para alcanzar mis sueños.

Gracias a la Universidad Estatal Península de Santa Elena por darme la oportunidad de pertenecer a la prestigiosa universidad y cada docente de la carrera de Telecomunicaciones por impartir sus conocimientos.

Mi gratitud a mi tutor, el Ing. Manuel Asdruval Montaño Blacio, por la compresión, tiempo, dedicación, paciencia y sobre todo por cada consejo brindado, un pilar fundamental en el desarrollo de este trabajo. Así mismo, al Ing. Daniel Jaramillo, por su compresión, sugerencias y aporte al proceso.

A mis compañeros Erick, Mae, Sebas, Andy y Heidy, que se convirtieron en mis amigos y fundamental durante este proceso, agradecida por su motivación, apoyo, ánimo y sobre todo su confianza. Gracias por su solidaridad y su apoyo en esos momentos difíciles de querer flaquear y abandonar la carrera.

Carmen Angelina Dominguez Silvestre

DECICATORIA

El presente trabajo de titulación se lo decido con todo mi corazón a mi mamá, Carmen Silvestre, por ese amor y sacrificio que han sido base para cada logro. Gracias por todo mamá por siempre estar a mi lado , por ese apoyo constantemente en toda mi vida y mi vida estudiantil por esas acompañadas en las noches de desvelo. Tus enseñanzas y tus esfuerzos se convirtieron en mi fuente de inspiración a seguir adelante y no flaquear.

A mi papá, Alcides Dominguez, gracias, papá por tu esfuerzo que hiciste y sigues haciendo por querer vernos superarnos cada día, por ayudarme en todo, por tu apoyo incondicional en los momentos que más te necesite, por cada palabra de aliento que me motivaron a seguir adelante.

Siempre estaré agradecida con ustedes por todo lo que han hecho por mí, por enseñarme que con esfuerzo se puede superar cada obstáculo. Este logro no solo es mío sino también de ustedes, porque sin su esfuerzo, dedicación y cariño, no habría sido posible cumplir este sueño.

A mis hermanas, Denisse y Sarita, quienes se convirtieron en mi mayor motivación de seguir adelante, gracias por su confianza hacia a mí, su apoyo incondicional y sobre todo estar en esos momentos difíciles. A mi sobrino Caleb, aunque hables en otro idioma gracias por quererme y amarme, eres mi gran motivo de seguir adelante. Gabriel, Ari y Abi gracias por su amistad y apoyo en todo tiempo. Los amo, los quiero y gracias por siempre estar a mi lado en esos momentos difíciles, gracias por todo.

Carmen Angelina Dominguez Silvestre

ÍNDICE GENERAL

ÍNDICE DE FIGURA

ÍNDICE DE ANEXO

RESUMEN

En el presente trabajo se desarrolló un sistema de seguridad interna basado en reconocimiento facial con Amazon Web Services (AWS). La importancia del sistema radica en que busca mejorar significativamente los sistemas convencionales de control de acceso a través de una solución biométrica.

El sistema permite la captura y procesamiento de imágenes a través de AWS Rekognition permitiendo validar la identidad en tiempo real, el servicio S3 para la gestión de almacenamiento de las imágenes encontradas atrás de la base de datos DynamoDB para gestionar los datos de las personas que serán validadas, el servicio Lambda para el procesamiento del código creado, para la respectiva comunicación se utiliza el servicio IoT Core conectando MQTT. El protocolo MQTT permite automatizar el control de acceso con una respuesta rápida y una comunicación eficiente. El desarrollo compone de la configuración del hardware, creación de bases de datos en la nube, integración de servicios de AWS, y la implementación de flujos en Node-RED para la gestión local mediante el acceso físico.

Los resultados demuestran una alta precisión del sistema, un alto grado de similitud de más del 95% para los usuarios registrados, una tasa de éxito en la implementación de más del 98% en AWS Lambda y un tiempo de respuesta promedio de menos de 2 segundos. El sistema cumple eficazmente con los objetivos establecidos, lo que confirma la posibilidad de utilizar plataformas inteligentes y tecnologías biométricas fáciles de implementar para garantizar la seguridad interna en un entorno controlado.

Palabras claves: Amazon Web Services (AWS), biometría, control de acceso, reconocimiento facial y seguridad interna.

ABSTRACT

In this work, an internal security system based on facial recognition was developed using Amazon Web Services (AWS). The importance of the system lies in its significant improvement over conventional access control systems through a biometric solution.

The system allows for the capture and processing of images through AWS Rekognition, enabling real-time identity validation. The S3 service is used to manage the storage of images found behind the DynamoDB database to manage the data of the individuals to be validated. The Lambda service is used to process the created code. The IoT Core service is used for communication via MQTT. The MQTT protocol allows for access control automation with rapid response and efficient communication. The development process includes hardware configuration, cloud database creation, AWS service integration, and the implementation of flows in Node-RED for local management via physical access.

The results demonstrate high system accuracy, a high degree of similarity of over 95% for registered users, a deployment success rate of over 98% on AWS Lambda, and an average response time of less than 2 seconds. The system effectively meets the stated objectives, confirming the feasibility of using smart platforms and easy-to-implement biometric technologies to ensure internal security in a controlled environment.

Keywords: Amazon Web Services (AWS), biometrics, access control, facial recognition, and internal security.

INTRODUCCIÓN

En la actualidad, la seguridad representa un gran reto en todo el mundo, tanto en lugares públicos como en privados. La sensación de inseguridad y el aumento de delitos han mostrado la importancia de utilizar tecnologías más modernas para asegurar un control de acceso efectivo, proteger a las personas y sus propiedades. Hay varios modos de seguridad tradicionales, como claves secretas, huellas o tarjetas que prometen protección dentro de diferentes entornos, están siendo vulnerables.

La forma de identificar a las personas por su rostro es una manera innovadora y eficaz, ayudando a reconocer rápidamente y siendo precisa, por sus características biométricas únicas. Este tipo de tecnología cuando se conjuga con plataformas inteligentes y servicios en la nube ofrece soluciones a escala alta de disponibilidad y con un nivel superior de fiabilidad.

El trabajo actual aparece como una solución a la necesidad de mejorar los sistemas de control de entrada usando un mecanismo para la seguridad interna que se basa en el reconocimiento de rostros, empleando objetos tecnológicos como Raspberry Pi, Amazon Web Services (AWS) y métodos claros y prácticos de comunicación como MQTT. Este sistema busca no solo aumentar la seguridad, sino también dar una opción barata que se pueda usar en variados ambientes, tales como oficinas, laboratorios, escuelas y lugares restringidos. Con esta idea se busca ayudar al progreso de soluciones de seguridad más innovadoras, que se manejan automáticas y sean confiables.

Esta propuesta busca ayudar a crear soluciones de seguridad más modernas, automáticas y confiables, que se adapten a las nuevas necesidades tecnológicas y sociales para proteger los espacios interiores.

CAPITULO I

1.1. Titulo:

Implementación de un sistema de seguridad interna basado en reconocimiento facial con plataforma de servicio inteligente

1.2. Descripción del Problema

Durante mucho tiempo, la seguridad ha sido un problema a nivel mundial generando temor y preocupación en algunas personas. Los ciudadanos cada vez más requieren que los gobiernos garanticen la seguridad, pero con frecuencia se ponen en duda su capacidad para lograrlo. En Ecuador, la seguridad ciudadana es un problema crítico, evidenciado por estadísticas que muestran un aumento en homicidios y denuncias por robos entre 2018 y 2019. Estos datos reflejan una creciente sensación de inseguridad y una mayor vulnerabilidad frente a la delincuencia[1]

La mayoría de las entidades no emplean sistemas de seguridad adecuados para prevenir el acceso no autorizado a sitios específicos. Utilizan herramientas como tarjetas magnéticas, códigos de seguridad y detección por huella dactilar, porque resulta ventajoso debido a su bajo costo de adquisición e implementación. Sin embargo, estos métodos conllevan un alto riesgo de vulnerabilidad, ya que existen varios factores que, en caso de darse, dificultarían la detección de una infracción a la seguridad[2].

Los espacios cerrados, tales como oficinas, laboratorios, almacenes, fábricas y escuelas, entre otros, están expuestos a diversos riesgos y amenazas, incluyendo robos, sabotajes o ataques internos. Estos espacios presentan desafíos únicos en términos de control de acceso,

vigilancia y detección de amenazas, y la seguridad debe mantenerse tanto en el interior como en el exterior de un lugar[3][4].

Los espacios internos enfrentan desafíos significativos, incluyendo la precisión del reconocimiento en condiciones variables de iluminación y ángulos, y la necesidad de integrar estos sistemas con las cámaras existentes para asegurar cobertura total. Es crucial desarrollar criterios claros para priorizar la vigilancia en áreas críticas dentro de los edificios, como oficinas sensibles, salas de servidores y entradas principales.

1.3. Antecedentes

El acceso a sistemas de seguridad avanzados suele ser costoso, lo que lo hace inaccesible para muchas personas. Por ello, se propone desarrollar una herramienta de control de acceso con las mismas capacidades que un sistema de alta gama, pero a un costo más accesible y utilizando software de código abierto. La gestión del personal en un establecimiento puede resultar complicada y difícil, lo que puede llevar a perder el control de la asistencia y plantear problemas de seguridad al no poder llevar un registro preciso de quién tiene acceso al establecimiento[5].

En [2] se desarrolló un prototipo para el control de accesos utilizando reconocimiento facial, con el objetivo de garantizar alta seguridad en el acceso a un sitio específico. El sistema combina sistemas biométricos convencionales y algoritmos inteligentes implementados en un software que opera en conjunto con el dispositivo LattePanda de alto rendimiento. Se estableció una base de datos en un servidor en la nube para la administración de la plataforma, permitiendo el acceso y monitoreo de datos en cualquier momento y lugar. Es un dispositivo con una aplicación fácil de usar que mejora la seguridad en la autenticación de usuarios.

En [6] se desarrolló un sistema para el registro automático de asistencia en clases, usando detección y reconocimiento facial con aprendizaje profundo. La arquitectura modular integró herramientas como Python con FastAPI y Django, junto con frameworks de Machine Learning y protocolos REST y MQTT. Se identifican individuos comparando vectores de características faciales almacenados en bases de datos, sin necesidad de reentrenar las redes neuronales al agregar nuevos datos. Modelos preentrenados de detección y reconocimiento facial, como MTCNN y FaceNet, fueron probados y evaluados para lograr alta precisión en el registro de asistencia.

En [7] se desarrolló un modelo de negocio para ofrecer servicios de seguridad a empresas retail en Ecuador, utilizando tecnología de reconocimiento facial y análisis de video con inteligencia artificial. Tras una investigación de mercado positiva, se implementó el servicio en Guayaquil, con planes de expansión. Se ofreció un paquete integral por $600 mensuales, incluyendo instalación, mantenimiento, monitoreo 24/7 y asesoría especializada. El análisis financiero demostró su viabilidad, con un VAN positivo de $44,156.61 y una TIR del 76.40%. La inversión inicial de $85,736 fue financiada en un 35% con capital propio y un 65% con financiamiento bancario.

1.4. Descripción del proyecto

El prototipo del sistema de seguridad interna basado en reconocimiento facial con una plataforma de servicio inteligente se realizará mediante el protocolo MQTT para conectar directamente con el dispositivo del hardware, hará la acción de capturar las imágenes que servirán como señal de entrada. Estas imágenes se enviarán a Amazon Web Services (AWS) para verificar si la persona tiene acceso autorizado.

Para la captura de datos, se usará un Raspberry Pi. El entrenamiento del modelo de reconocimiento facial se realizará en una plataforma en la nube, donde se subirá y gestionará la base de datos necesaria para este propósito. El entrenamiento de la red neuronal se llevará a cabo utilizando herramientas avanzadas de aprendizaje automático en la nube, lo que permitirá actualizar y optimizar continuamente el modelo para asegurar un sistema de seguridad preciso y confiable.

1.5. Objetivos del proyecto

1.5.1. Objetivos generales

Desarrollar un sistema de seguridad basado en reconocimiento facial mediante plataforma de servicio inteligente para mejorar la eficiencia y protección de espacios internos.

1.5.2. Objetivos específicos

- Realizar un estado del arte sobre tecnologías y algoritmos de reconocimiento facial, incluyendo sus aplicaciones y metodologías de implementación.
- Seleccionar y adquirir los recursos de hardware y software necesarios para la implementación del sistema de reconocimiento facial, teniendo en cuenta las características técnicas.
- Configurar y entrenar el algoritmo de visión por computador adaptándolo a las necesidades del entorno, considerando factores como la precisión.
- Realizar pruebas exhaustivas del sistema implementado para evaluar su precisión y eficacia, frente a diferentes escenarios de campo.

1.6. Justificación

El prototipo del sistema de seguridad interna basado en reconocimiento facial con una plataforma de servicio inteligente se justifica por su capacidad para mejorar significativamente la seguridad mediante la identificación rápida y precisa de individuos. Este sistema aumenta la eficiencia operativa al automatizar los procesos de identificación, reduce costos a largo plazo al eliminar la necesidad de tarjetas de acceso físicas y ofrece una personalización y escalabilidad que se adapta a las necesidades específicas de cada organización.

Un sistema de seguridad ofrece múltiples beneficios, proporcionando una mayor sensación de seguridad en las personas, garantizando un control eficiente al momento de acceder a diferentes áreas específicas.

El reconocimiento facial se fundamenta en algoritmos sofisticados que permiten analizar y comparar grandes conjuntos de datos visuales en tiempo real. Esto posibilita una detección precisa y rápida de rostros, incluso en diferentes ángulos de visión. Es una tecnología que ofrece un amplio abanico de aplicaciones en diversos campos, desde la seguridad y vigilancia hasta la identificación de usuarios en dispositivos móviles.

Los sistemas de reconocimiento facial emplean plataformas inteligentes se pueden ampliar fácilmente y son adaptables. Esto significa que dichos sistemas pueden adaptarse a las necesidades específicas de cualquier organización e implementarse en diversos entornos, aumentando así la flexibilidad para la gestión de la seguridad interna y permitiendo la expansión del sistema ante cambios o crecimiento de los requisitos de seguridad con facilidad en el futuro.

1.7. Alcance

El alcance de este proyecto es entregar un sistema de seguridad interna de reconocimiento facial previamente entrenado. Además, se configurará una plataforma en la nube para gestionar y actualizar el modelo. Toda esta configuración y entrenamiento forman parte integral del proyecto.

El dispositivo permitirá la captura de imágenes, el procesamiento local de los datos y la verificación en tiempo real mediante la conexión a la nube. Esto asegurará un nivel de precisión y seguridad en el acceso a un lugar. Con el prototipo, se proporcionará un control de acceso eficiente y fiable.

CAPITULO II

2.1. Marco Contextual

La seguridad interna en entornos cerrados, tales como oficinas y escuelas, es esencial para la protección de los individuos y bienes dentro de estos espacios. En este contexto, la implementación de tecnología de reconocimiento facial se presenta como una herramienta avanzada y eficaz. Esta tecnología emplea las características únicas del rostro humano para identificar a las personas, reduciendo así el riesgo de accesos no autorizados.

El reconocimiento facial se integra de manera efectiva con el Internet de las Cosas (IoT). Esta integración permite la conexión de dispositivos de captura de datos y procesamiento local con servicios en la nube, facilitando el análisis de datos en tiempo real. De esta manera, se crea un sistema de seguridad que no solo es robusto, sino también dinámico y adaptable a diferentes escenarios.

El sistema propuesto busca ser una solución accesible y eficiente, asegurando un alto nivel de seguridad sin comprometer la facilidad de uso. Con el avance continuo de las tecnologías de reconocimiento facial y la expansión del IoT, se espera que estas soluciones se conviertan en el estándar para la seguridad interna en entornos cerrados, proporcionando una mayor tranquilidad y protección a todos los usuarios.

2.2. Marco Teórico

2.2.1. Seguridad en espacios institucionales

Las instalaciones educativas deben garantizar la seguridad para proteger a los docentes, estudiantes y personal administrativo. Se enfrentan a muchos riesgos, incluidos robos y

actividades delictivas, por lo que implementar medidas preventivas es esencial para abordar y reducir estas amenazas[8].

2.2.1.1. Espacios Internos

Las áreas de protección dentro de las instituciones educativas son esenciales para garantizar la seguridad y el bienestar de todos los miembros de la comunidad escolar, incluyendo estudiantes, maestros, personal administrativo y visitantes. Estos espacios, que abarcan desde zonas de recreo y aulas hasta laboratorios y oficinas, están diseñados para minimizar riesgos y proporcionar un ambiente seguro y saludable[9].

Medidas de seguridad en espacios internos

- **Control de acceso:** Limitar el acceso a las secciones internas de la escuela al personal autorizado mediante la implementación de un lector de tarjetas o un sistema biométrico.
- **Videovigilancia:** Instalar cámaras de seguridad estratégicamente en todo el recinto escolar para observar la actividad en áreas críticas, como los pasillos de entrada y las zonas de alta seguridad.
- **Alarmas de seguridad:** Instalar alarmas de seguridad para notificar a las autoridades en caso de emergencia, permitiendo una respuesta rápida y precisa.
- **Personal de seguridad:** Asignar personal capacitado para supervisar y abordar emergencias.
- **Equipos de detección:** Utilizar detectores de movimiento para identificar intrusiones y notificar al personal de seguridad[10].

2.2.1.2. Espacios Externos

Son áreas como patios, jardines, áreas de juego y estacionamientos fuera de las instituciones educativas. Estos espacios juegan un papel importante en el crecimiento y desarrollo integral de los estudiantes y, por tanto, su diseño y uso deben ser abordados para mejorar el aprendizaje y el bienestar[9].

Medidas de seguridad en espacios externos

- **Perímetro seguro**: Verificar y asegurar los límites de la escuela para evitar el ingreso de personas no autorizadas.
- **Puntos de acceso controlados**: Regular los puntos de entrada y salida, estableciendo restricciones de acceso para visitantes y vendedores del personal estudiantil.
- **Videovigilancia exterior**: Instalar cámaras de seguridad fuera de la estructura escolar, especialmente en las entradas y salidas, para monitorear el movimiento y los disturbios en las áreas circundantes.
- **Acceso de visitantes**: Administrar y registrar las llegadas de visitantes y proveedores para garantizar que solo las personas autorizadas ingresen al recinto escolar.
- **Monitoreo de áreas de estacionamiento**: Monitorear los espacios de estacionamiento para disuadir entradas no autorizadas y garantizar la seguridad de los vehículos[10].

2.2.1.3. Importancia de la seguridad en los espacios institucionales

La seguridad en las instituciones educativas es fundamental para proteger a los estudiantes y al personal, establecer un entorno favorable para el aprendizaje y preservar la reputación de la institución. Los padres desean que los centros educativos sean lugares seguros para sus hijos. Por otro lado, una institución educativa con deficiencias de seguridad puede ver afectada su reputación y su capacidad para atraer estudiantes y personal[8].

2.2.2. Biometría

La biometría se deriva de las palabras "bio", que significa vida, y "metría", que significa medida. Mediante su uso, se identifican las características de las personas, generalmente fisiológicas, como las huellas digitales, el reconocimiento de iris, la geometría de la mano, entre otras, para la verificación de su identidad. No obstante, la aplicación más habitual de la biometría ha sido en los sistemas de seguridad, con el propósito de detectar y registrar el acceso de individuos a un sitio específico[11].

Sistemas biométricos

El objetivo de los sistemas biométricos es recopilar, almacenar y comparar datos, específicamente las características biológicas de una persona, de manera automática y confiable. Dado que utilizan patrones individuales únicos e irrepetibles, son ampliamente utilizados en los sistemas de acceso. Los patrones se reconocen en tres etapas[2]:

- La etapa inicial permite el uso de sensores para la toma de datos y la definición de patrones.
- La segunda etapa implica la extracción de datos fundamentales para la creación de una base de datos.

- La tercera y última etapa permite que los patrones se clasifiquen según las clases.

2.2.2.1. Tipos de Biometría

Los tipos de sistemas biométricos varían según las características biológicas a evaluar. Como resultado, se pueden dividir en dos categorías:

- Sistemas biométricos que se basan en rasgos fisiológicos.
- Sistemas biométricos que se basan en el comportamiento.

Los sistemas biométricos basados en características fisiológicas, también conocidos como estáticos, utilizan características fisiológicas distintivas e irrepetibles como la huella dactilar, la retina, el rostro y el iris, entre otros. Por otro lado, existen sistemas biométricos basados en el comportamiento, también conocidos como dinámicos, que evalúan las características generadas por una acción en particular, como la voz humana, los movimientos corporales y los gestos, entre otras[2].

2.2.2.1.1. Reconocimiento de huella dactilar.

Los sistemas biométricos que utilizan el reconocimiento de huellas dactilares se encuentran entre los más utilizados en la práctica. Las huellas dactilares son rasgos físicos únicos de cada individuo, lo que permite una identificación precisa y segura. Los sistemas biométricos que utilizan la identificación por huellas dactilares se utilizan ampliamente. Las huellas dactilares, como características físicas distintas de cada persona, permiten un reconocimiento preciso y seguro[12].

El registro de huellas dactilares implica recopilar y almacenar la huella digital de un usuario. Al iniciar sesión, el sistema compara la huella digital del usuario con la almacenada. Sin

embargo, surge un problema importante ya que este sistema puede generar huellas dactilares duplicadas incluso si los usuarios cooperan, lo que permite a los atacantes recopilar huellas dactilares para acceder al sistema sin autorización[12].

Para evitar este tipo de ataques, se empleó métodos como verificar la legitimidad de la muestra (diferenciar entre huellas dactilares auténticas y copiadas) y proteger los datos biométricos transferidos[12].

2.2.2.1.2. Reconocimiento del iris

El sistema biométrico de reconocimiento del iris utiliza un método de reconocimiento de patrones para analizar la imagen de alta calidad del iris en los ojos de las personas. Lleva a cabo la autenticación capturando fotografías detalladas del iris con cámaras infrarrojas especiales, que ayudan a reducir los reflejos de la córnea para mostrar los detalles complejos del iris. Estos detalles se procesan para identificar características únicas, como patrones y estructuras de color, y luego se almacenan como plantillas digitales[13].

2.2.2.1.3. Reconocimiento de retina

El sistema de reconocimiento por retina es uno de los métodos de identificación biométrica que utiliza una estructura única para autenticar a una persona. Para este tipo de análisis, se realiza un escaneo de la retina y se analiza la capa de vasos sanguíneos en la parte trasera del ojo[13].

En cuanto al funcionamiento, se emplea una fuente de luz de baja intensidad y un acoplador óptico que lee los patrones específicos con gran precisión. Las personas que utilizan lentes deben quitárselos, ya que es necesario ubicar los ojos cerca del dispositivo implementado

y enfocarlos en un punto determinado. El usuario debe fijar la vista en una pequeña luz para que el aparato pueda escanear y establecer la identidad de la persona[13].

2.2.2.1.4. Reconocimiento facial.

El sistema de reconocimiento facial emplea algoritmos avanzados para la identificación y verificación de la identidad de una persona mediante el análisis de su rostro. Este proceso es una combinación de varias etapas técnicas y complejas, cada una desempeñando un papel crucial para garantizar la precisión y la fiabilidad del sistema[14].

2.2.2.1.5. Reconocimiento de voz con redes neuronales.

Este sistema tiene la capacidad de aprender y generalizar diferentes patrones en señales de audio. Las redes neuronales son utilizadas para reconocer patrones en el tiempo, la frecuencia y la amplitud de la señal de audio. Gracias a esto, es posible identificar las características de la persona que está hablando[15].

La arquitectura de las redes neuronales empleadas en este sistema incluye redes neuronales de Kohonen, redes neuronales profundas y redes neuronales recurrentes. A continuación, se detallan las etapas de preprocesamiento y entrenamiento[15].

2.2.3. Reconocimiento Facial

Cuando desconocemos la identidad de una persona, utilizamos un sistema de reconocimiento facial para averiguarlo. Este sistema nos ayuda a realizar tareas como iniciar sesión en nuestras cuentas y acceder a información importante. Para determinar la identidad de alguien, se buscan imágenes en una base de datos y, si se encuentra una coincidencia, se revela la identidad de la persona[16].

El principal objetivo de un sistema de reconocimiento facial es identificar rápidamente a una persona en una imagen o en un vídeo, determinando si se trata de una persona o de algo distinto. El reconocimiento facial puede llevarse a cabo de dos maneras: observando el aspecto de alguien, basado en su apariencia, o utilizando un modelo especial para identificarlo, basado en algoritmos específicos[16].

2.2.3.1. Ventajas de reconocimiento facial

Seguridad eficiente

El reconocimiento facial es un método rápido y confiable para verificar la identidad de una persona. A diferencia de los escáneres de huellas dactilares o de retina, el reconocimiento del iris es más veloz y fácil de usar. Es más sencillo identificar rostros que recordar contraseñas o PIN. Para mantener las cuentas seguras, es recomendable habilitar la autenticación multifactorial, lo que incrementa la seguridad[16].

Mayor precisión

El uso del reconocimiento facial es una forma más precisa de identificar a las personas en comparación con el uso de un número de teléfono móvil, una dirección de correo electrónico, una dirección postal o una dirección IP. Muchos servicios, como la compra de acciones o el comercio de criptomonedas, emplean el reconocimiento facial para proteger tanto a los clientes como a su dinero[16].

Una integración más fácil

La mayoría de los programas de software de seguridad pueden operar con tecnología de reconocimiento facial, lo que la convierte en una adición conveniente y perfecta para los sistemas de seguridad. Por ejemplo, los teléfonos inteligentes están equipados con cámaras

avanzadas que pueden reconocer rostros o con software especial para realizar funciones interesantes[16].

2.2.3.2. Funcionamiento de reconocimiento facial

La tecnología de reconocimiento facial ha avanzado significativamente, permitiendo diversas formas de mapear rostros y almacenar datos faciales, cada una con su propio nivel de precisión y eficiencia. En este contexto, se puede explorar cómo funciona el reconocimiento facial, cómo se almacenan los datos y quién puede acceder a ellos.

Existen tres formas principales de reconocer rostros:

- **Reconocimiento facial tradicional:** Permite identificar a una persona con solo observar una imagen.
- **Reconocimiento facial 3D:** Esta tecnología utiliza escáneres especiales para tomar fotografías de personas desde todos los ángulos, generando una representación tridimensional de su rostro.
- **Reconocimiento facial biométrico:** Funciona de manera similar a un detective muy inteligente, capaz de identificar a una persona con solo mirar su rostro[14].

2.2.3.3. Métodos tradicionales de reconocimiento facial

El reconocimiento facial tradicional se puede dividir en dos tipos principales: holístico y basado en características.

- **Reconocimiento facial holístico:** Este método adopta un enfoque integral, examinando cada aspecto del rostro de una persona para encontrar la combinación perfecta.

- **Reconocimiento facial basado en características:** Este método es similar a un detective que busca pistas importantes en un rostro y luego las compara con el rostro de un sospechoso para ver si coinciden[14].

2.2.4. Reconocimiento facial 3D

Para obtener un reconocimiento facial más preciso, se utilizan sensores especiales que pueden capturar la forma del rostro en 3D. A diferencia del reconocimiento facial tradicional, el reconocimiento facial 3D no se ve afectado por la iluminación y puede funcionar incluso en la oscuridad. Este método permite el reconocimiento facial incluso si la persona no está mirando directamente a la cámara, proporcionando una capacidad de reconocimiento desde diferentes ángulos.

El iPhone X y versiones más recientes incorporan Face ID, una tecnología que utiliza el reconocimiento facial 3D para identificar al usuario. El proceso de reconocimiento facial 3D consta de seis pasos importantes[14].

2.2.5. Reconocimiento facial biométrico

La biometría de la piel y el rostro es un área nueva y emocionante en la tecnología de reconocimiento facial que la hace aún más precisa. El análisis de la textura de la piel se centra en un área específica mediante un sofisticado algoritmo que mide líneas, texturas y poros con gran detalle[14].

2.2.5.1. Almacenamiento de los datos de reconocimiento facial

Los datos de reconocimiento facial probablemente se encuentren en su bolsillo o en su mano en este momento. Muchos teléfonos inteligentes admiten el inicio de sesión biométrico mediante huellas dactilares y reconocimiento facial. Debido a que los datos biométricos se

almacenan en el propio dispositivo, estos pueden verificarse sin necesidad de enviarlos a otra fuente.

Sin embargo, aunque es poco probable que las huellas faciales de Apple y Samsung se encuentren en una base de datos extensa, es posible que así sea. Una gran cantidad de agencias gubernamentales poseen bases de datos de reconocimiento facial. El FBI, por ejemplo, tiene una base de datos en expansión en los Estados Unidos que contiene más de 640 millones de imágenes. Recientemente, China hizo que los escaneos faciales sean obligatorios para aquellos que deseen obtener un plan de datos móviles, y también mantiene una base de datos de reconocimiento facial para identificar o verificar a casi 1,400 millones de sus ciudadanos[14].

A diferencia de las bases de datos privadas u oficiales, las bases de datos públicas de reconocimiento facial ofrecen una gran cantidad de imágenes gratuitas. Empresas de corredores de datos, como Clearview AI, tienen una base de datos con más de tres mil millones de imágenes extraídas de las redes sociales, registros de empleo y sitios de noticias en la web. Este es uno de los desarrollos más preocupantes en las bases de datos de reconocimiento facial[14].

Los corredores de datos no solo recopilan fotos. Los anunciantes monitorean la mayoría de las actividades en línea a través de cookies, incluyendo lo que se ve, se compra y los sitios web que se visitan. Esa información se combina con frecuencia con los datos de la huella digital del navegador, una forma más profunda de seguimiento en línea que utiliza scripts ocultos para identificar al usuario mediante las características distintivas de su navegador y equipo[14].

La configuración del navegador, el idioma y la zona horaria, las extensiones instaladas, el sistema operativo, la tarjeta gráfica y muchas otras características pueden utilizarse para crear una huella digital del navegador que sea muy precisa para identificar al usuario.

Utilizando un navegador privado y seguro que incluya tecnología de bloqueo del reconocimiento, se puede entorpecer y evitar este tipo de recopilación de datos. Avast Secure Browser ha sido especialmente diseñado para evitar rastreadores web intrusivos, confundir los scripts de huellas digitales del navegador y bloquear los anuncios. Se recomienda probar este navegador seguro y gratuito para disfrutar de una experiencia web verdaderamente privada[14].

2.2.5.2. La precisión de reconocimiento facial

En condiciones ideales, los algoritmos de reconocimiento facial presentan una precisión casi perfecta. En entornos controlados, la tasa de éxito es mayor; sin embargo, en el mundo real, el rendimiento suele ser inferior. Dado que ninguna medida proporciona una visión completa, es complicado predecir con exactitud la tasa de éxito de esta tecnología[16].

Por ejemplo, los algoritmos de verificación facial, que buscan coincidencias entre personas e imágenes de referencia claras, como una licencia de conducir o una fotografía policial, logran puntuaciones de alta precisión. Sin embargo, solo bajo las siguientes condiciones se alcanza este nivel de precisión:

- Iluminación y ubicación coherentes
- Rasgos faciales limpios y sin obstrucciones
- Fondos y colores controlados
- Resolución de la imagen y calidad de la cámara adecuadas

El envejecimiento es otro factor que influye en las tasas de error. Los rostros cambian con el tiempo, lo que dificulta su comparación con fotografías tomadas años antes[16].

2.2.5.3. Seguridad del reconocimiento facial

Los sistemas de reconocimiento facial utilizan patrones matemáticos únicos para almacenar datos biométricos, convirtiéndose así en uno de los métodos más seguros y efectivos de identificación biométrica. Para prevenir accesos no autorizados, los datos faciales pueden ser anonimizados y mantenidos en privado. Además, la tecnología de detección de actividad vital permite distinguir entre usuarios vivos y simples imágenes faciales, lo que evita que el sistema sea engañado por una fotografía de un usuario activo[16].

2.2.6. Detección de rostros

La detección facial es una tecnología de visión computarizada que localiza y mide los rostros humanos en imágenes o videos. Pertenece al subtipo de detección de objetos o clases, cuya tarea consiste en encontrar la ubicación y tamaño de los objetos dentro de una imagen perteneciente a una clase específica.

Aunque esta tarea es sencilla para los humanos, las computadoras pueden enfrentar dificultades debido a varios factores, como la variabilidad en la posición del rostro, la presencia o ausencia de componentes estructurales como el bigote, expresiones faciales, oclusiones como el uso de lentes o gorros, y condiciones ambientales.

El primer paso en un sistema de reconocimiento facial es detectar el rostro humano, cuyo resultado puede variar dependiendo de si se utiliza un conjunto de imágenes estáticas o video en tiempo real. Por lo tanto, debe ser capaz de identificar rostros independientemente de estos factores.

Existen cuatro categorías de técnicas de detección facial, no excluyentes entre sí:

1. **Metodologías basadas en el conocimiento:** Utilizan distancias y posiciones entre características humanas como los ojos, la nariz y los labios, codificando el conocimiento humano.
2. **Métodos basados en rasgos invariantes:** Detectan características que no cambian con la luz, la pose o la ubicación de la cámara, como la ceja, la nariz, la textura de la piel y la línea del cabello. Estos métodos crean modelos estadísticos a partir de uno de estos componentes y luego verifican los resultados.
3. **Métodos basados en moldes:** Establecen una conexión entre una imagen de entrada y un patrón previamente definido para capturar características faciales.
4. **Técnicas basadas en apariencia:** Utilizan modelos obtenidos a través del entrenamiento de imágenes, considerando la imagen como un vector de características. Los patrones se determinan mediante el aprendizaje de imágenes, en contraste con los métodos basados en moldes que utilizan un patrón definido por un experto.

2.2.7. Amazon Web Services (AWS)

2.2.7.1. AWS Rekognition

AWS Rekognition ofrece un servicio de reconocimiento de imágenes basado en Deep Learning (Aprendizaje Profundo). Este sistema tiene la capacidad de detectar escenas, rostros y objetos, reconocer personas, extraer texto y detectar contenido inapropiado en diferentes imágenes. Además, permite buscar y comparar rostros. Todo este proceso se lleva a cabo gracias a la tecnología de Deep Learning que Amazon emplea en Prime Photos[17].

AWS Rekognition cuenta con una puntuación de fiabilidad al momento de identificar elementos, lo que ayuda en la toma de decisiones. Cuando un rostro es detectado, se ubica en posiciones de coordenadas (x, y, w, h), permitiendo localizar su posición en la imagen. Esta plataforma también ofrece una API, por lo que no es necesario tener conocimiento de Deep Learning, facilitando la interacción con el servicio de manera sencilla[17].

2.2.7.2. Desarrollo de AWS Rekognition

AWS Recognition es un servicio público de bajo costo, utilizado por numerosos usuarios a nivel mundial. Se realizará un análisis previo del servicio utilizando ejemplos identificados con la implementación de Facenet, que está incluido en este servicio. Además, se enfocará en las predicciones obtenidas con altas probabilidades. Este servicio ofrece diversas funcionalidades que ayudarán en la evaluación de este trabajo, ya que estará centrado en la comparación de rostros contra una base de datos integrada en el sistema[17].

Conjuntos de datos

Para la recopilación de datos, se utilizará el mismo conjunto que se empleó para entrenar el modelo Facenet. Este conjunto incluirá los elementos necesarios para el entrenamiento y las pruebas. En el momento de la recopilación, la identificación de rostros ya estará guardada y no será necesario utilizar Rekognition, ya que proporcionará la funcionalidad requerida[17].

El API se usará para la identificación de rostros, proporcionando información sobre la persona identificada, como su edad, si lleva gafas, si tiene bigote y un análisis de sentimientos. Una vez analizadas todas las imágenes, estas se almacenarán en un bucket de S3[17].

Pre-procesamiento de conjunto de datos

Este servicio aceptará imágenes en formato PNG y JPEG con un tamaño máximo de 15 MB, no requerirá preprocesamiento adicional. Además, se debe considerar un tamaño mínimo de imagen de 80x80 píxeles, pero al menos 40x40 píxeles deben ocupar el rostro de la persona. En el caso de imágenes con múltiples rostros, se utilizará el rostro más grande para compararlo con todos los rostros almacenados en la base de datos con el fin de identificar la persona. Todas las imágenes en el conjunto de datos ya están recortadas a una resolución de 160 por 160 píxeles, las cuales serán utilizadas para proporcionar la información necesaria y fortalecer las predicciones[17].

Entrenamiento

Para la parte de entrenamiento, no es necesario entrenar un modelo, pero se debe definir lo que se conoce como "Colección". Esta será creada para almacenar la información de cada uno de los rostros detectados. El documento proporcionado por AWS menciona que todas las imágenes no se almacenan en diferentes colecciones, sino que se procesan mediante un algoritmo de detección de rostros. Este algoritmo extrae los atributos de un rostro en un vector de características (feature vector). Una vez extraídos, estos atributos se almacenan en la colección, AWS no especifica el algoritmo utilizado solo da a conocer que se basa en redes neuronales convolucionales (CNN), FaceNet y Deep Learning. [17].

Cada colección tiene la capacidad de almacenar hasta 20 millones de rostros. Cada una de estas colecciones estará asociada a una versión específica del modelo utilizado para el reconocimiento facial. Por defecto, al crear una nueva colección, esta se asociará a la última versión del modelo disponible. No es posible migrar una colección a una nueva versión del modelo directamente, ya que las diferentes versiones no son compatibles entre sí. Sin embargo,

existe una manera de migrar de una versión del modelo a otra reindexando todas las imágenes, por lo que es crucial mantener las imágenes almacenadas[17].

2.2.7.2.1. Límites de la solución

Resolución de imagen: Amazon Rekognition es capaz de procesar imágenes en diversas resoluciones. Para obtener resultados óptimos, se recomienda utilizar una resolución VGA (640x480) o superior. Si la resolución es inferior a QVGA, es probable que no se puedan identificar rostros, objetos o contenidos inapropiados.

Rostros por imagen: Amazon Rekognition puede identificar hasta 100 rostros en una sola imagen.

Cantidad de rostros por collection: Cada colección tiene capacidad para almacenar hasta 20 millones de imágenes faciales[18].

2.2.7.3. Amazon Simple Storage Service (S3)

Amazon Simple Storage Service (S3) es un servicio de almacenamiento en la nube ofrecido por Amazon Web Services (AWS). Este servicio se destaca por sus altos estándares de escalabilidad, disponibilidad y seguridad. S3 puede ser utilizado para aplicaciones, sitios web, usos empresariales, respaldos de seguridad y big data.

S3 es una solución de almacenamiento en la nube que permite a los usuarios almacenar y recuperar cualquier cantidad de datos en cualquier momento y desde cualquier lugar. Este servicio es extremadamente escalable y flexible, lo que lo convierte en una opción ideal para empresas de todos los tamaños que necesitan gestionar grandes volúmenes de datos de manera eficiente. A continuación, se describen en detalle las características y funcionalidades de Amazon S3(ver figura 1)[19].

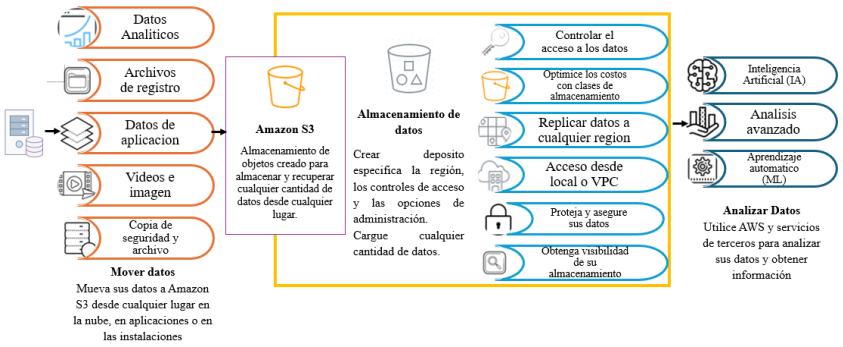

Figura 1. Entrenamiento de S3 en Amazon Web Services[19]

2.2.7.4. Amazon Simple Queue Service (SQS)

Amazon Simple Queue Service (SQS) proporciona una cola para desacoplar procesos grandes, permitiendo almacenar en buffers y procesar los datos en lotes hasta que toda la información sea procesada. Este servicio permite enviar, almacenar y recibir mensajes entre software desde cualquier lugar sin perder mensajes, y sin requerir otros servicios adicionales. A continuación, se describen en detalle las características y funcionalidades de Amazon Simple Queue Service (ver figura 2)[20].

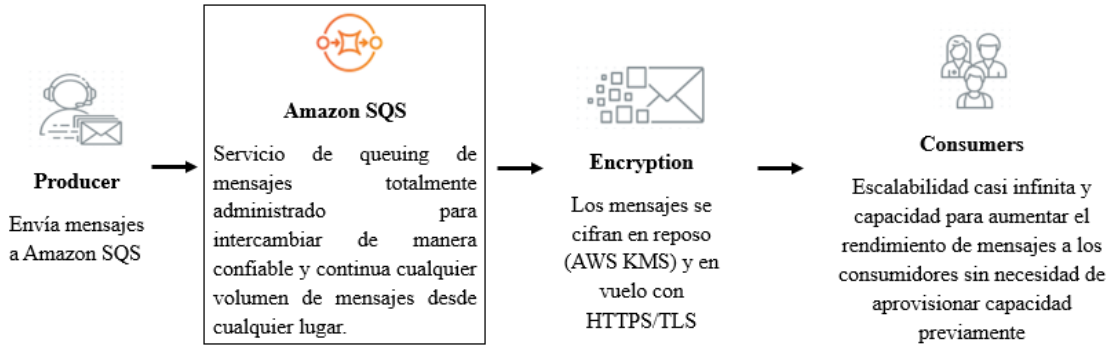

Figura 2. Entrenamiento SQS en Amazon Web Services [20]

2.2.7.5. Amazon Simple Notification Service (SNS)

Amazon Simple Notification Service (SNS) proporciona una gran variedad de opciones para enviar notificaciones, incluyendo A2A (Application-to-Application) y A2P (Application-to-Person). SNS ofrece mensajería de alto rendimiento basada en un modelo push, que puede integrarse con sistemas como microservicios y aplicaciones distribuidas sin necesidad de un servidor centralizado. Entre las aplicaciones mencionadas se incluyen Amazon SQS, Amazon Kinesis Data Firehose, AWS Lambda y múltiples puntos de conexión HTTP[21].

La funcionalidad de las notificaciones A2P permite enviar mensajes a clientes a través de mensajes de texto, notificaciones push o correos electrónicos[21]. A continuación, se describen en detalle las características y funcionalidades de Amazon Simple Notification Service (ver figura 3, 4 y 5).

a) Publicación/Suscripción

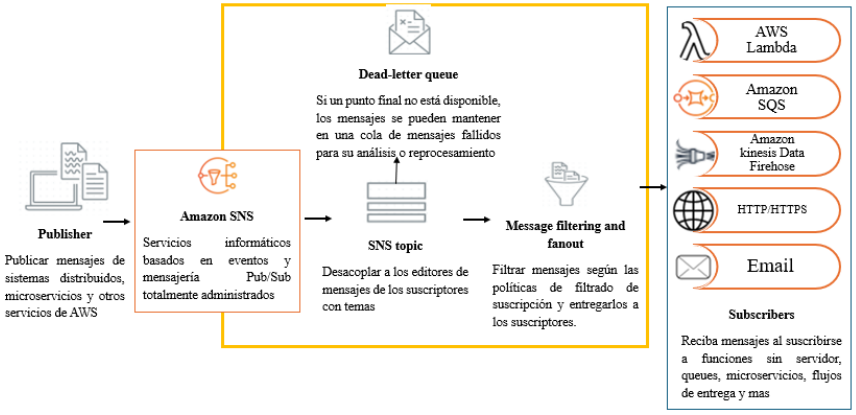

Figura 3. Entrenamiento de SNS – Publicación/Suscripción [21]

b) SMS

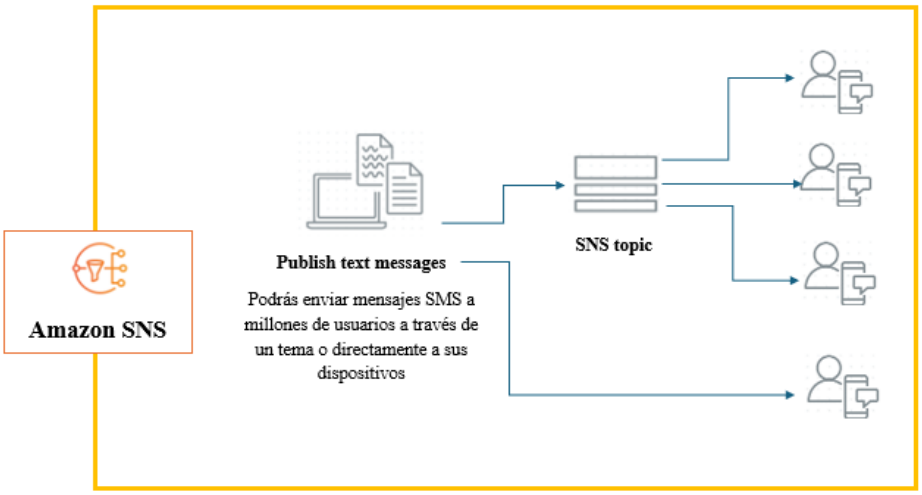

Figura 4. Entrenamiento de SNS – SMS [21]

c) Notificaciones push móviles

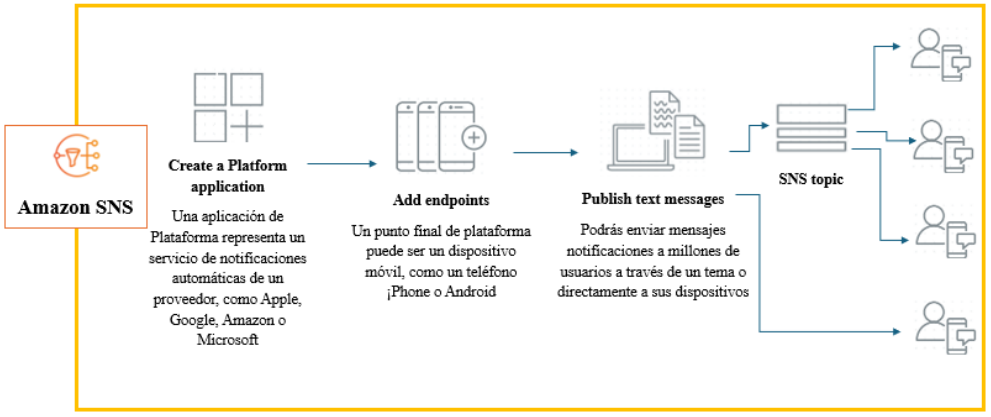

Figura 5. Entrenamiento de SNS – Notificaciones push móviles [21]

2.2.7.6.Amazon API Gateway

Es un servicio para crear, publicar, monitorear y proteger las API REST, HTTP en diferentes escalas. Se crean API para tener acceso a servicios web como datos almacenados en la nube, para el uso en las aplicaciones del cliente de AWS y para aplicaciones externas[22].

Crea API REST que se basa en HTTP, habilita comunicación entre clientes y servidores sin estados e implementan métodos HTTP con sus estándares como POST, PUT, PATCH, GET y DELETE. Crea API de WebSocket permite comunicación entre el servidor y el cliente en dúplex completo y dirigen los mensajes entrantes en función del contenido[22].

Arquitectura de API Gateway

Los clientes desarrollarán una experiencia de desarrollador integrado y coherente al momento de crear aplicaciones sin servidores de AWS, gestiona tareas relacionadas con el procesamiento de llamadas simultáneas al API. Administra el tráfico, el acceso y autorización. Este servicio actúa como una puerta principal para que las aplicaciones puedan acceder a los datos, a los backends, trabajos de Amazon EC2, códigos que se ejecutan en AWS Lambda y aplicaciones web[22](ver figura 6).

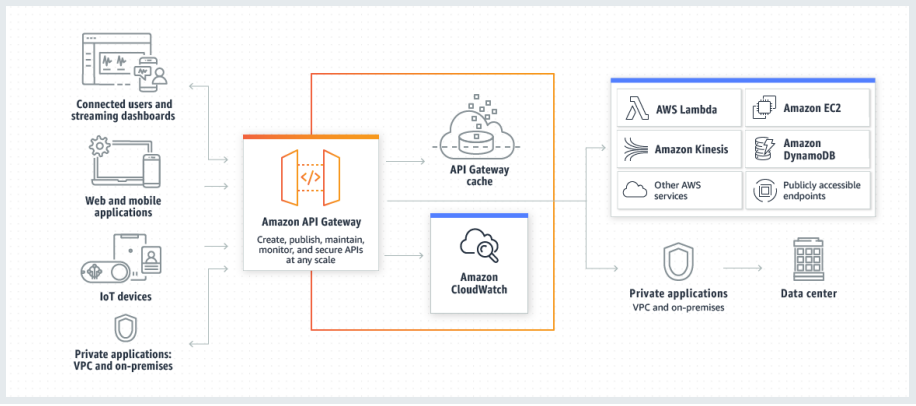

Figura 6. Arquitectura de AWS Lambda [22]

2.2.7.7.AWS Lambda

Es un servicio basado en eventos y no tiene necesidad de tener un servidor, para las aplicaciones web se utiliza un paradigma de programación. El usuario es libre de organizar y escribir el código en función de Lambda que es uno de los componentes principales al momento de crear una función, controla el acceso por medio de los permisos Lambda y administra

automáticamente los roles de ejecución con los servicios. Lambda proporciona diferentes lenguajes de programación y el tiempo de ejecución[23].

Lo procesos que se pueden utilizar en Lambda es el procesamiento de flujos, Aplicaciones Web, backends móviles, backends de IoT, procesamientos de archivos, operaciones e integración de bases de datos y tareas programadas y periódicas[23].

2.2.7.8. DynamoDB

DynamoDB está diseñada para optimizar las cargas de trabajo al momento que requiere de un rendimiento a alta escala. El servicio permite tener un rendimiento de milisegundos a un digito en cualquier escala, totalmente administrado, trabaja sin servidor y como base utiliza NoSQL. NoSQL ofrece un mejor rendimiento, capacidad de administrar, flexibilidad y escalabilidad[24].

2.2.7.9. AWS IoT Core

Permite conectarse con dispositivos IoT, servicios, aplicaciones para enviar o recibir datos de AWS IoT Core a otros dispositivos da opción a controlar y administrar dichos dispositivos con una solución eficiente, puede interactuar con puntos de conexiones planos y puntos finales de LoRa WAN[25].

Los puntos de conexión del plano del control, puntos finales del dispositivos y puertas de enlace LoRaWAN permiten tener acceso a puntos de conexión, autenticación y SDK con las herramientas proporcionada, cada punto de acceso tiene finalidad, formato y servicio diferentes[25](ver figura 7).

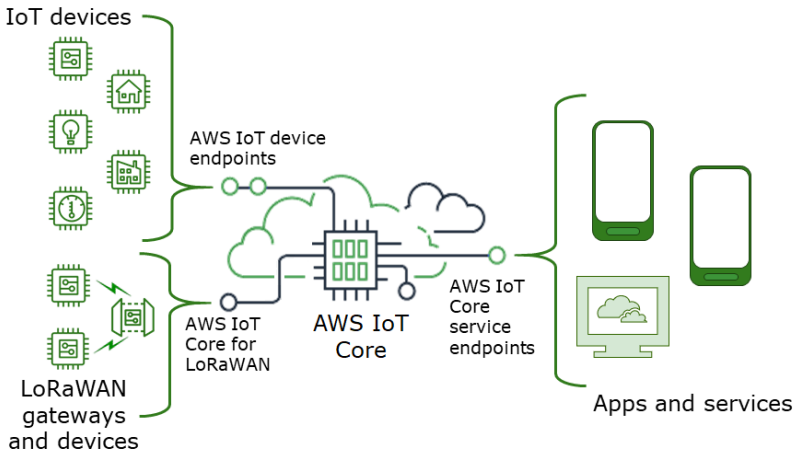

Figura 7. Arquitectura de IoT Core [25]

2.2.8. Google Cloud Platform (GCP)

El líder de buscadores Google nos brinda la solución de computación en la nube llamada Google Cloud Platform. Además de los servidores, este espacio contiene más de 90 servicios que han ayudado al éxito de Google Cloud Platform y que pueden ayudarnos a expandir nuestro negocio[26].

2.2.8.1. Servicios de Google Cloud Platform

Compute Engine

La oferta de infraestructura como servicio (IaaS) de Google Compute Engine (GCE) permite a los clientes ejecutar cargas de trabajo en el hardware físico de Google. Para ese propósito, Google Compute Engine proporciona una cantidad escalable de máquinas virtuales (VM) que funcionan como grandes clústeres informáticos. Es posible administrar GCE a través de una API RESTful, una GUI de línea de comandos o una consola web. El precio de Compute Engine se basa en el pago por uso, que se cobra por segundo con un mínimo de un minuto[27].

Cloud Storage

El almacenamiento en nube es una forma de almacenamiento de datos de computadora en la que los datos digitales se almacenan en servidores fuera de las instalaciones. Los servidores son mantenidos por un proveedor externo que se encarga de alojar, administrar y proteger los datos almacenados en su infraestructura. El proveedor garantiza que las conexiones a Internet públicas o privadas siempre estén disponibles para los datos en sus servidores. Las organizaciones pueden almacenar, acceder y mantener datos con Cloud Storage, lo que les permite cambiar de un modelo de gastos de capital a un modelo de gastos operativos. La escalabilidad de Cloud Storage permite a las organizaciones aumentar o reducir su huella de datos según sea necesario[28].

Kubernetes Engine

Google Kubernetes es un servicio que permite la ejecución y la supervisión de aplicaciones en contenedores utilizando Google Cloud Platform. GKE se basa en Kubernetes, un sistema de orquestación de contenedores de código abierto que le permite implementar y administrar aplicaciones en contenedores de gran tamaño. GKE ofrece una variedad de funciones que facilitan la implementación, ejecución y administración de aplicaciones en contenedores en Google Cloud Platform. Examinemos algunas características y desventajas. Para facilitar la administración y el descubrimiento, organiza los contenedores que componen una aplicación en unidades lógicas. Kubernetes se basa en las mejores ideas y prácticas de la comunidad, junto con 15 años de experiencia en la ejecución de cargas de trabajo de producción en Google[29].

Cloud SQL

Cloud SQL ofrece bases de datos relacionales totalmente gestionadas y es un servicio fácil de usar. Cloud SQL permite que Google se encargue de las tareas rutinarias pero necesarias pero que con frecuencia requieren mucho tiempo, como la aplicación de parches y actualizaciones, la configuración de copias de seguridad y la gestión de copias de seguridad, para que pueda concentrarse en la creación de grandes aplicaciones. Cada instancia de Cloud SQL incluye un cortafuegos de red, que le permite administrar el acceso de la red a su instancia de base de datos otorgando permisos. Cloud SQL es simple de usar y no requiere instalación ni mantenimiento de software[30].

Auto ML

El auto aprendizaje (AutoML) se ha popularizado tanto en el sector como en el campo de la investigación académica de inteligencia artificial (IA). Por su capacidad para generar resultados explicables y reproducibles, AutoML tiene un gran potencial para proporcionar soluciones de IA en sectores regulados. AutoML facilita el desarrollo de IA para personas sin conocimientos teóricos de ciencia de datos. Los expertos en aprendizaje automático deben realizar manualmente cada paso del proceso actual de creación de prototipos en ciencia de datos, como el preprocesamiento de datos, la ingeniería de características y la optimización de hiperparámetros[31].

Dialog Flow

Dialogflow es una plataforma con comprensión del lenguaje natural que facilita el diseño de interfaces de usuario de conversación para dispositivos móviles, aplicaciones web, bots, sistemas de respuesta de voz interactiva y otros dispositivos. Dialogflow le permite crear formas

nuevas y atractivas para que los usuarios interactúen con su producto. Dialogflow puede analizar entradas de texto o audio de tus clientes. Además, puede responder a sus clientes de una variedad de maneras, como texto o voz sintética[32].

Big Query

BigQuery es un almacén de datos de bajo costo, multinube, escalable, sin servidores y totalmente administrado de Google Cloud. Incluye herramientas de análisis que pueden procesar petabytes de datos en segundos, todo de manera económica y adaptable a las necesidades de cada organización. BigQuery es una herramienta muy poderosa debido a su facilidad de uso, su capacidad para manejar grandes cantidades de datos y su integración con herramientas de aprendizaje automático e inteligencia artificial. Los datos se almacenan en tablas que se pueden organizar por proyectos y líneas de negocios. Google BigQuery fue desarrollado para mejorar las consultas de datos, lo que lo diferencia de otros almacenes de datos de GCP[33].

2.2.8.2. Cuando utilizar GCP

Google Cloud Platform es una plataforma ideal para materializar las ideas de innovación de tu equipo de trabajo al contar con herramientas dedicadas para esto, por lo que es excelente para empresas que sean apasionados por el desarrollo de sus propias aplicaciones. Además, funciona de manera excelente con herramientas de Open Source que facilitan la tarea y, sobre todo, hacen que una aplicación sea económica[34].

Instructivo del proceso de reconocimiento facial

Crea el objetivo de servicio

Para acceder a las API de Google con los SDK cliente oficiales, crea un objeto de servicio basado en el documento de descubrimiento de las API, que describe la API al SDK. Tendrás que recuperarlo desde el servicio de descubrimiento de la API de Vision[34].

Envía una solicitud de detección de rostro

A fin de construir una solicitud para la API de Vision, primero debes consultar la documentación de la API. En este caso, le pedirás al recurso images que haga una operación annotate en tu imagen. Una solicitud a esta API adopta la forma de un objeto con una lista de requests. Todos los elementos de esta lista incluyen dos tipos de información:

- Los datos de la imagen codificados en base64
- Una lista de características para las que querrías tener anotaciones respecto de la imagen[34].

Procesa la respuesta

La respuesta a nuestra solicitud de anotación de rostros contiene un conjunto de metadatos sobre los rostros detectados, que incluyen coordenadas de un polígono que abarca el rostro. Sin embargo, en este punto, solo es una lista de números. Los usaremos para confirmar que efectivamente encontraste los rostros en tu imagen. Dibujaremos polígonos en una copia de la imagen mediante las coordenadas que muestra la API de Vision[34]. A continuación, se describen en detalle las características y funcionalidades de Google Cloud Platform (ver figura 8)**.**

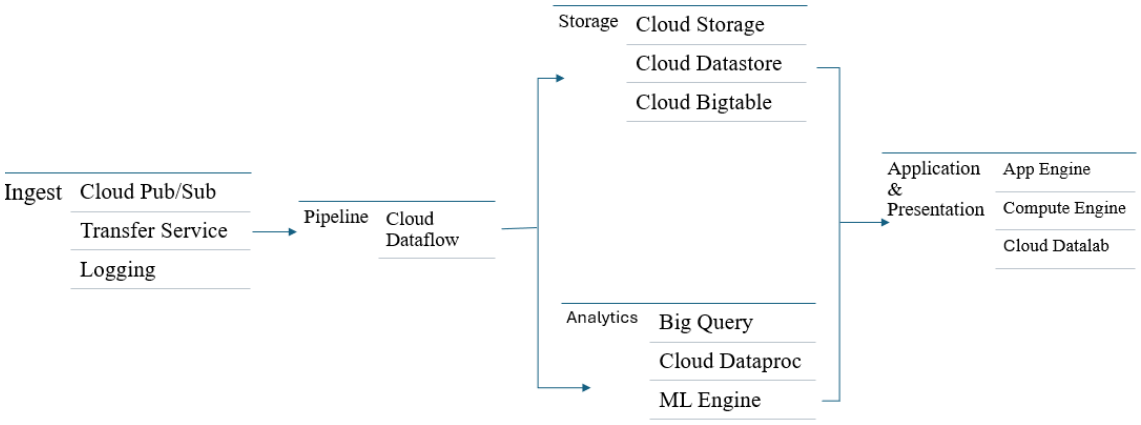

Figura 8. Proceso de entrenamiento en Google Cloud Platform (GCP) [34]

2.2.9. Protocolo MQTT

2.2.9.1. Importancia del MQTT

Debido a sus ventajas, el protocolo MQTT se ha convertido en un estándar para la transmisión de datos en el Internet de las Cosas.

Ligero y efectivo

La implementación de MQTT en dispositivos IoT requiere recursos mínimos, lo que permite su uso incluso en microcontroladores de tamaño reducido. Un mensaje de control MQTT puede contener tan solo dos bytes de datos. Además, los encabezados de los mensajes MQTT son pequeños, maximizando así el ancho de banda de la red[35].

Escalable

La implementación de MQTT requiere una cantidad mínima de código y las operaciones demandan muy poca energía. El protocolo incluye funciones que permiten la comunicación con una gran cantidad de dispositivos IoT, posibilitando la conexión con millones de estos dispositivos mediante MQTT[35].

Fiable

Muchos dispositivos IoT utilizan redes celulares poco confiables con bajo ancho de banda y alta latencia. MQTT incorpora funciones que reducen el tiempo de reconexión de un dispositivo IoT con la nube. Además, ofrece tres niveles de calidad de servicio para garantizar la fiabilidad en los casos de uso de IoT: al máximo (0), al menos (1) y exactamente una vez (2)[35].

Seguro

MQTT permite a los desarrolladores autenticar usuarios y dispositivos mediante protocolos de autenticación modernos como OAuth, TLS 1.3 y certificados administrados por el cliente[35].

Admitido

Varios lenguajes de programación, como Python, están ampliamente disponibles para implementar el protocolo MQTT. Esto permite a los desarrolladores utilizarlo rápidamente en cualquier tipo de aplicación con una codificación mínima[35].

2.2.9.2. En que se basa MQTT

El protocolo MQTT se basa en el modelo de suscripción o publicación. Los clientes y los servidores se comunican directamente entre sí en la comunicación de red tradicional. Los clientes solicitan al servidor recursos o datos, y el servidor procesa y envía una respuesta. Sin embargo, MQTT desconecta al editor o remitente del mensaje del receptor del mensaje mediante el uso de un patrón de publicación o suscripción. En cambio, un tercer componente, conocido como agente de mensajes, tiene el control sobre la comunicación entre editores y suscriptores. El trabajo del agente es filtrar todos los mensajes entrantes de los editores y distribuirlos a los suscriptores de manera adecuada[35].

2.2.9.3. Componentes MQTT

La siguiente ilustración muestra cómo MQTT implementa el modelo de publicación o suscripción.

Clientes de MQTT

Un cliente MQTT es cualquier dispositivo, desde servidores hasta microcontroladores, que ejecuta una biblioteca MQTT. El cliente actúa como editor y receptor si envía mensajes. En esencia, cualquier dispositivo que se comunique a través de una red mediante MQTT se denomina dispositivo cliente MQTT.

Agentes de MQTT

El sistema de back-end MQTT coordina los mensajes entre varios clientes. Los deberes del agente incluyen recibir y filtrar mensajes, identificar a los clientes suscritos a cada mensaje y enviarles mensajes. Además, se encarga de tareas adicionales como:

- Los clientes autorizan y validan MQTT
- Transferir mensajes a otros sistemas para su evaluación posterior
- Controlar las sesiones de clientes y los mensajes perdidos[35].

Conexión MQTT

Una conexión MQTT permite que los agentes y los clientes se comuniquen. Los clientes envían un mensaje CONECTAR al agente MQTT para iniciar la conexión. Al responder a un mensaje CONNACK, el agente confirma la conexión. El agente y el cliente MQTT requieren una pila TCP o IP para comunicarse. Los clientes solo hablan con el agente y nunca se comunican entre sí[35].

2.2.9.4. Funcionamiento del MQTT

A continuación, se proporciona una descripción general del funcionamiento de MQTT:

- Un cliente MQTT se conecta al agente MQTT.
- Una vez conectado, el cliente puede publicar mensajes, suscribirse a mensajes particulares o realizar ambas acciones.
- El agente MQTT, al recibir un mensaje, lo envía a los suscriptores interesados[35].

2.2.9.5. Implementar MQTT en Amazon Web Services (AWS)

AWS IoT Core es un servicio completamente administrado que permite la conexión de miles de millones de dispositivos IoT y el enrutamiento de miles de millones de mensajes a los servicios de AWS. Este servicio ofrece las siguientes capacidades:

- Conectar, administrar y escalar flotas de dispositivos de manera fácil y fiable, sin necesidad de aprovisionar ni administrar servidores.
- Elegir el protocolo de comunicación preferido, incluidos MQTT, HTTPS, MQTT a través de WSS y LoRaWAN.
- Proteger las conexiones y los datos de los dispositivos mediante autenticación mutua y cifrado de extremo a extremo.
- Filtrar y transformar los datos de los dispositivos y actuar sobre ellos en tiempo real, basándose en reglas empresariales definidas[35].

CAPITULO III

3.1. Componentes de la propuesta

Para la elaboración de esta propuesta tecnológica, se requerirán componentes físicos y componentes lógicos, los cuales se detallarán a continuación.

3.1.1. Componentes físicos

Para la implementación del sistema de seguridad interna, se utilizarán los siguientes dispositivos electrónicos que permitirán realizar el proceso de detección mediante visión por computadora.

3.1.1.1. Raspberry pi 4 Model B de 8 GB

El Raspberry Pi 4 Model B (ver figura 9) es un miniordenador debido a su arquitectura completa, que integra componentes esenciales, accesible por su bajo costo en comparación a computadoras tradicionales lo convierte en una solución accesible[36].

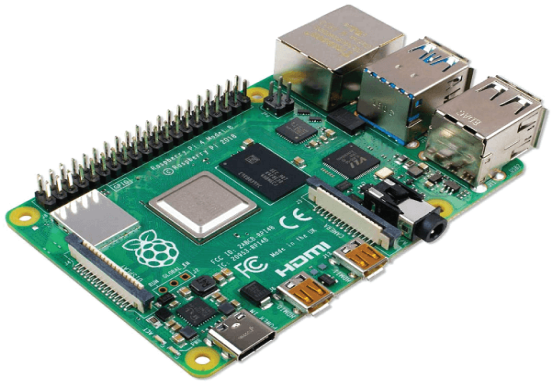

Figura 9. Raspberry Pi 4 Model B de 8GB [36]

3.1.2. Componentes lógicos

3.1.2.1. Plataforma de servicio inteligente (Amazon Web Services)

Amazon Web Services pone a su disposición una plataforma de servicios en las que ofrece Inteligencia Artificial, Machine Learning e IoT son solo algunas de las herramientas más complejas que pueden ser aprovechadas. Resuelve problemas de seguridad, eficiencia y tiempo de respuesta convirtiéndolos en soluciones en tiempo real; su infraestructura brinda confiabilidad respecto al rendimiento ofrecido.

3.1.2.2. Software Raspberry Pi Imager

Raspberry Pi Imager es un software que permite realizar la instalación de sistemas operativos en tarjetas microSD para su uso en dispositivos Raspberry Pi. Es una manera sencilla de poder descargar e instalar el OS para el Raspberry Pi a través de esta herramienta.

3.1.2.3.Node-RED

Node-RED es una herramienta de programación visual que facilita la programación, permitiendo a las personas conectar nodos para procesar datos y moverlos entre dispositivos y servicios. Es útil porque ofrece nodos preconfigurados que manejan eficientemente los mensajes MQTT.

3.2. Diseño de la propuesta

3.2.1. Diagrama funcional del sistema

En esta sesión se detallará el funcionamiento de todo el sistema que se llevará a cabo en este proyecto mediante diagramas de flujo, para eso se divide en 2 etapas.

- Etapa de registro en la base de datos de Amazon Web Services (AWS).

- Etapa de control de acceso mediante la tecnología de reconocimiento facial.

3.2.1.1. Etapa de registro en la base de datos de Amazon Web Services (AWS).

Se inicia sesión utilizando las credenciales de AWS del usuario, mediante comandos ejecutados en la consola (cmd), se crea una colección en Amazon Rekognition. Posteriormente, también a través de comandos, se sube la fotografía correspondiente, asegurándose de copiar el faceId generado y de que la imagen esté correctamente almacenada en Amazon S3. En la base de datos DynamoDB se registra la información del usuario, incluyendo Nombres, Apellidos, Cédula y el faceId, lo que resulta fundamental para vincular los datos biométricos con la identidad de cada persona(ver figura 10).

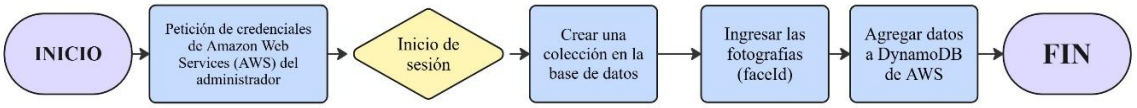

Figura 10. Proceso de registro en DB de AWS

3.2.1.2. Etapa de control de acceso mediante la tecnología de reconocimiento facial.

El sistema solo podrá avanzar y capturar la imagen si se tiene acceso a una cámara, de otro modo, no será posible tomar una fotografía. Una vez que la captura esté lista, será enviada por medio del API para que sea analizada y procesada en Amazon Rekognition. Cuando la captura es procesada en Amazon Rekognition emite una respuesta encontrada dentro de la base de datos en la que esa respuesta será trasmitida en formato JSON a Node-RED por medio del protocolo MQTT.

En Node-RED se realiza un procesamiento adicional encargado de analiza el grado de similitud de la captura y al comparar los puntos obtenidos, si es mayor al 95%, el sistema lanza un mensaje de que ha sido verificado la persona, por lo que da apertura a la puerta. En caso

contrario, que no ha sido verificado la identidad de quien desea ingresar no tendrá apertura(ver figura 11).

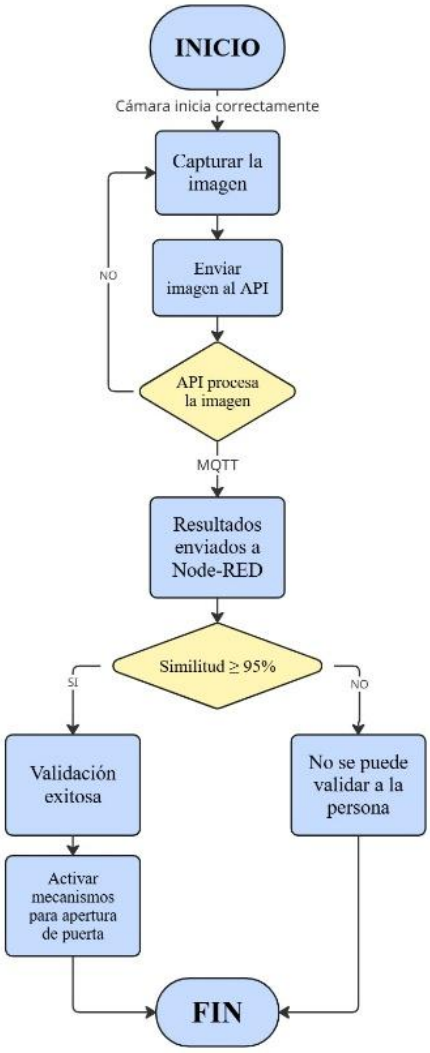

Figura 11. Proceso de control de acceso por reconocimiento facial con AWS

3.2.2. Diseño físico del sistema de seguridad interno basado en reconocimiento facial con Amazon Web Services

El proceso inicia escaneando el código QR e ingresando a la interfaz de reconocimiento facial teniendo conexión a internet y dando permiso al inicio de la cámara en el dispositivo, luego capturando el rostro y enviándolo a verificar por medio del API Gateway en la que es

procesada por la función Lambda. Esta función utiliza un servicio que analiza la imagen con Amazon Rekognition para comparar el rostro capturado con los datos almacenados en la base de datos.

El resultado de reconocimiento facial es enviado desde la función Lambda a IoT Core, en la que va a transmitir por MQTT al Raspberry que estará conectado a una cerradura eléctrica en la puerta para conceder o denegar el acceso. El sistema permite automatizar en tiempo real y comunicación con dispositivos IoT en una arquitectura escalable y segura en la nube(ver figura 12).

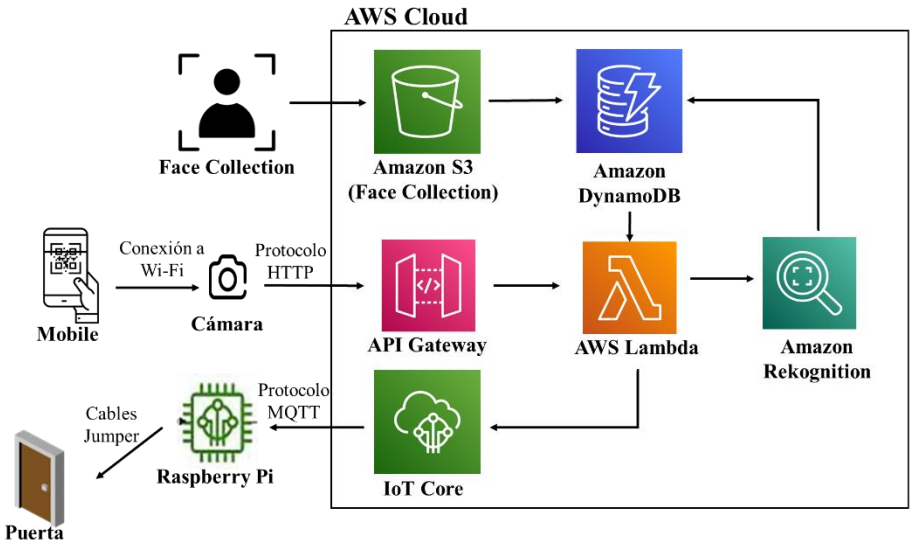

Figura 12. Diseño físico del sistema de seguridad basado reconocimiento facial con AWS

3.2.3. Diseño electrónico

Para un sistema de control de seguridad se utiliza un Raspberry Pi, un módulo de relé de un canal y una cerradura eléctrica. El Raspberry Pi actuará como un controlador, enviando una señal desde uno de sus pines GPIO al módulo de relé, hará la acción de activar o desactivar la cerradura mediante una fuente de alimentación adecuada. Se hace para que la corriente de la cerradura no vaya directo al Raspberry Pi.

En el circuito se incluye un diodo (flyback) para proteger el relé y el Raspberry Pi, conectando en paralelo con el terminal de la cerradura, el diodo hará disipar los picos de voltaje (ver figura 13).

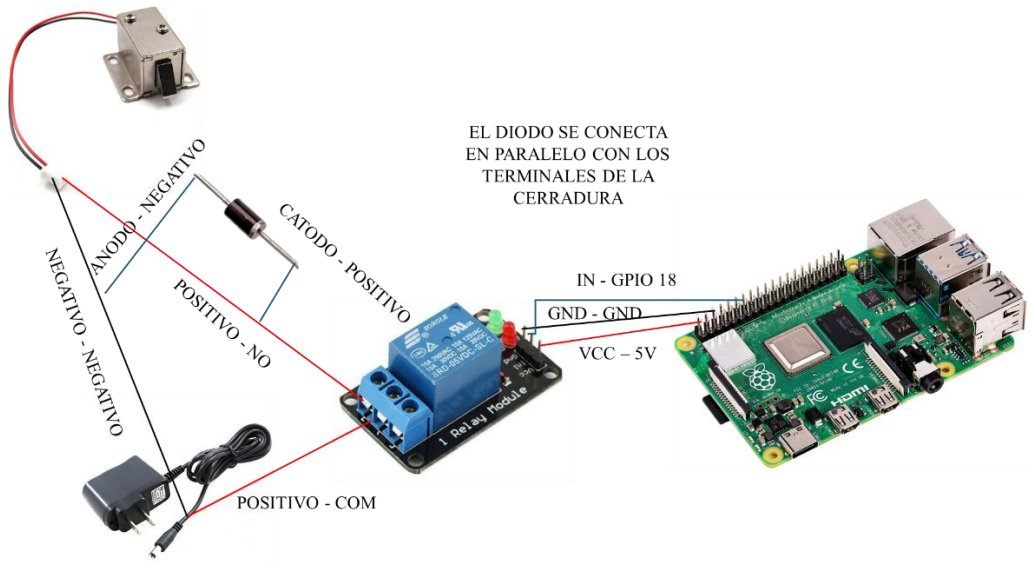

Figura 13 Diseño eléctrico para la acción de la puerta

3.3. Desarrollo del diseño de la propuesta

3.3.1. Configuración del OS para Raspberry pi 4

Para la configuración el usuario debe descargar Raspberry Pi Imager en la página oficial **https://www.raspberrypi.com/software/**, hacer clic en **"Download for Windows"**. Una vez descargado se tiene que seguir ciertos pasos para instalar correctamente, el paso a paso para la instalación se encuentra en el siguiente enlace **https://raspberrypi.cl/2021/05/13/instalacion-de-sistema-operativo-con-pi-imager/.**

Si prefiere una ruta más ágil, el usuario también puede abrir el terminal e ingresar el comando **"sudo apt install rpi-imager"** para instalar la herramienta (ver figura 14).

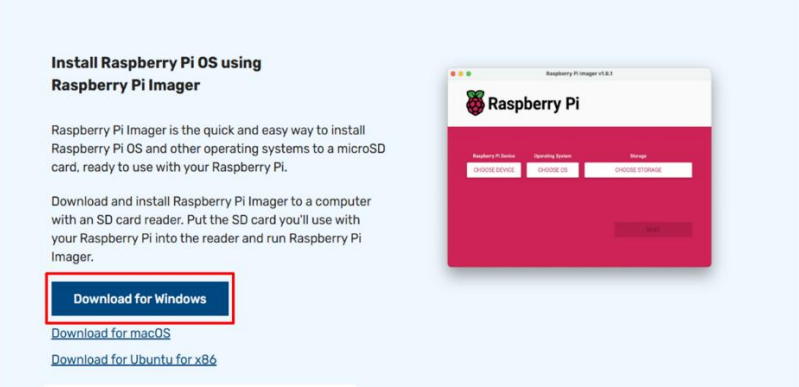

Figura 14. Descargar Rasberry Pi Imager

Cuando se termina de escribir todo el sistema operativo en el pendrive, se retira la tarjeta SD para insertar el puerto SD del Raspberry Pi. Luego, se conecta la fuente de alimentación y, con un cable HDMI – Tipo B, se conecta a una pantalla. Se espera unos minutos y aparece el escritorio (ver figura 15).

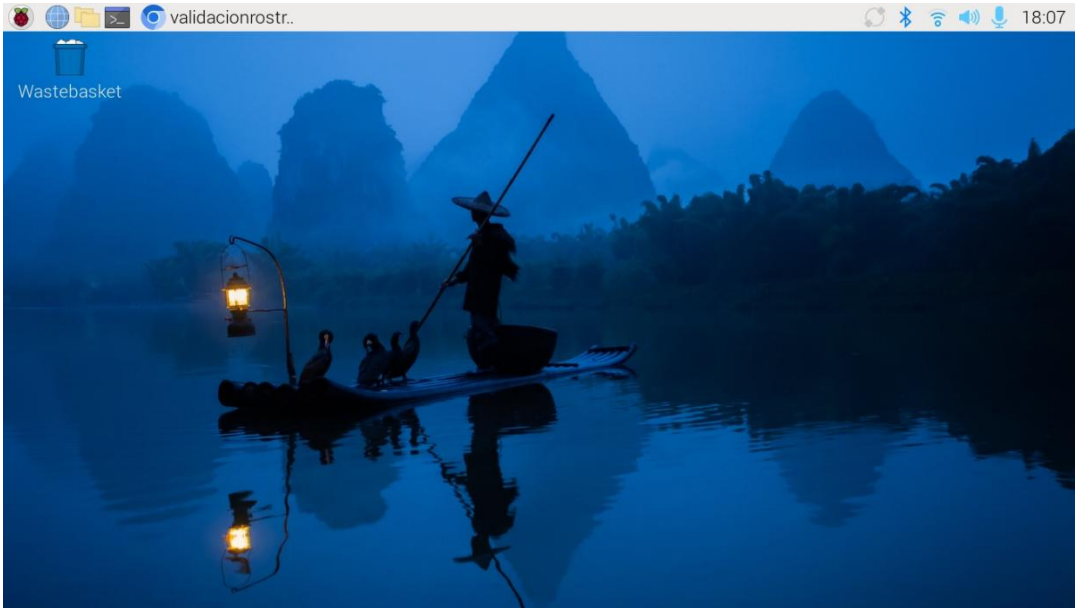

Figura 15. Entorno del escritorio del Raspberry Pi

3.3.2. Configuración en Amazon Web Services (AWS)

3.3.2.1. Creación de la cuenta Amazon Web Services

Para crear una cuenta de AWS, se elige una dirección de correo electrónico es preferible que sea personal y asignar un nombre para la respectiva cuenta. Cuando se completan los campos con la información, se da clic en "**Verify email address"**(ver figura 16)**.** En las que se recibe un correo para la respectiva verificación del correo electrónico.

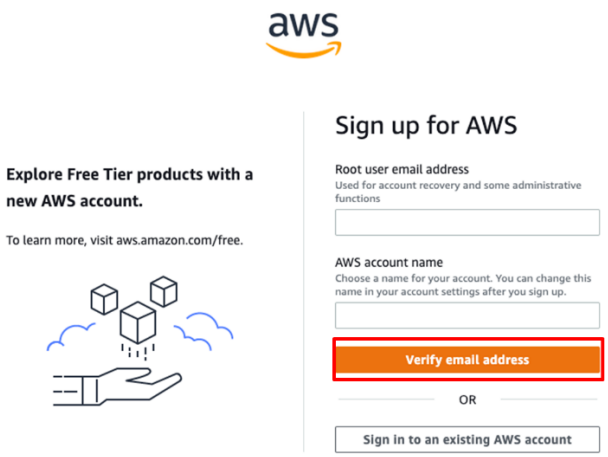

Figura 16. Creación de la cuenta AWS - Verificar correo electrónico

Después de la respectiva verificación se crea una contraseña para el usuario raíz. Tomar en cuenta que la contraseña debe incluir: letras minúsculas, mayúscula, números y caracteres. Luego de confirmar la contraseña se da clic en **"Continuar"** (ver figura 17)**.**

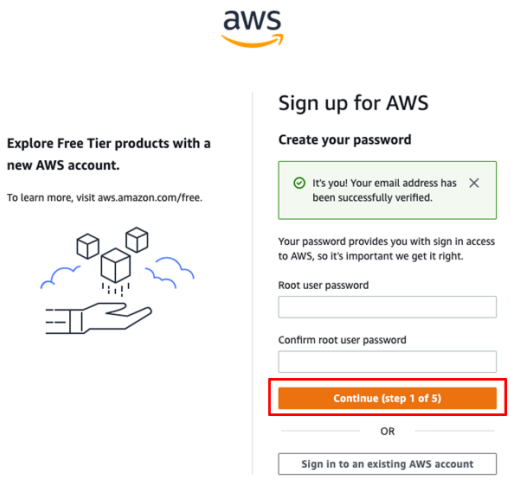

Figura 17. Creación de la cuenta AWS - Crear contraseña

En la siguiente pantalla se requiere la información del contacto y como se va a utilizar AWS. Dentro de los datos solicitados se debe proporcionar la información de una cuenta con fines de facturación y así abrir la cuenta, no olvidar hacer clic en la casilla para aceptar los términos y condiciones de AWS. Después de terminar de llenar el formulario hacer clic en **"Continuar"** (ver figura 18).

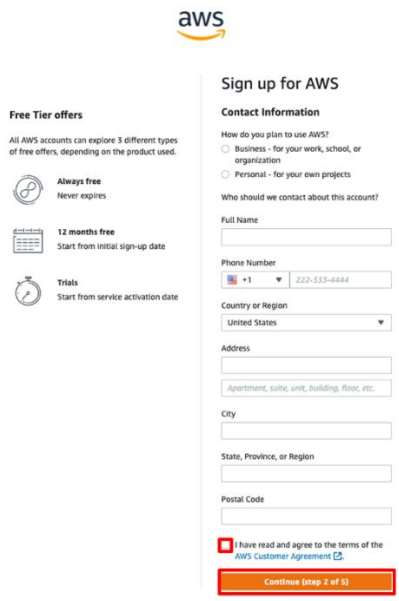

Figura 18. Creación de la cuenta AWS - Agregar información del usuario para verificar la cuenta

Para la verificación de la cuenta se debe agregar la información de una tarjeta de crédito o débito. Cuando se haya completado con la información dar clic en **"Verificar y Continuar"** (ver figura 19).

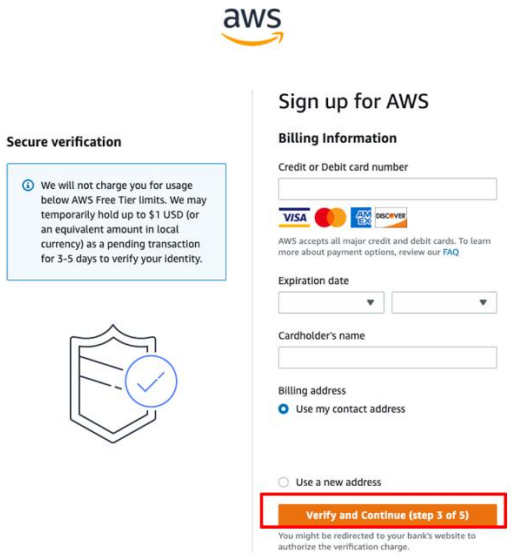

Figura 19. Creación de la cuenta AWS – Incorporar una tarjeta de débito para hacer la respectiva facturación

La cuenta se verifica por medio de mensaje de texto (SMS) y llamada de voz en las se enviará un código y así poder ingresar el código correcto y hacer clic en **"Send SMS" figura** (ver figura 20).

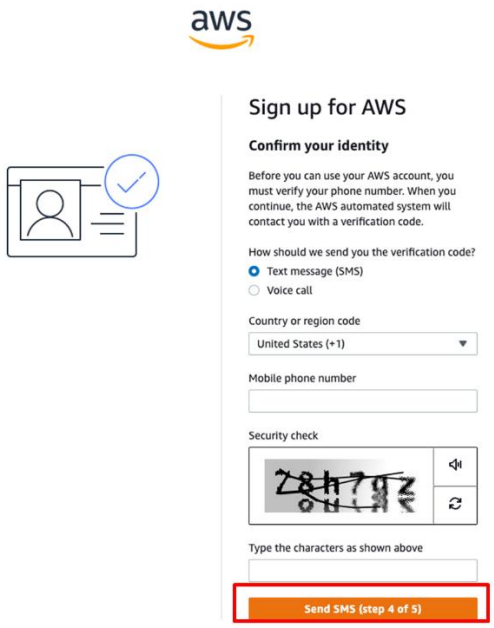

Figura 20. Creación de la cuenta AWS - Verificar la cuenta por SMS o llamada de voz

Para el último paso se elige el plan de soporte para la cuenta de AWS. En las que AWS tiene 3 planes la más utilizada se llama Basic Support y es gratuita. Para terminar, hacer clic en **"Complete sign up"** (ver figura 21).

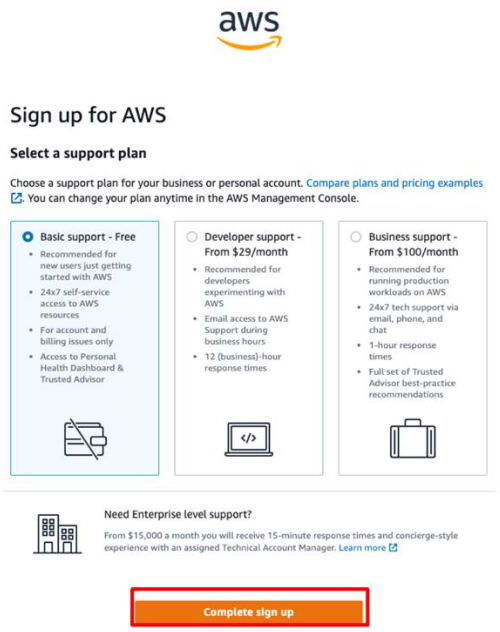

Figura 21. Creación de la cuenta AWS - Elegir el plan a utilizar en la plataforma

Después de terminar de crear la cuenta se observa la consola de administración de AWS (Amazon Web Services), en las que se encuentra **Recently visited** donde se puede observar la lista a las se accedido recientemente por otro lado se observa el apartado de **Application**s en donde se puede crear nuevas aplicaciones dando clic en **"Create Application"** (ver figura 22).

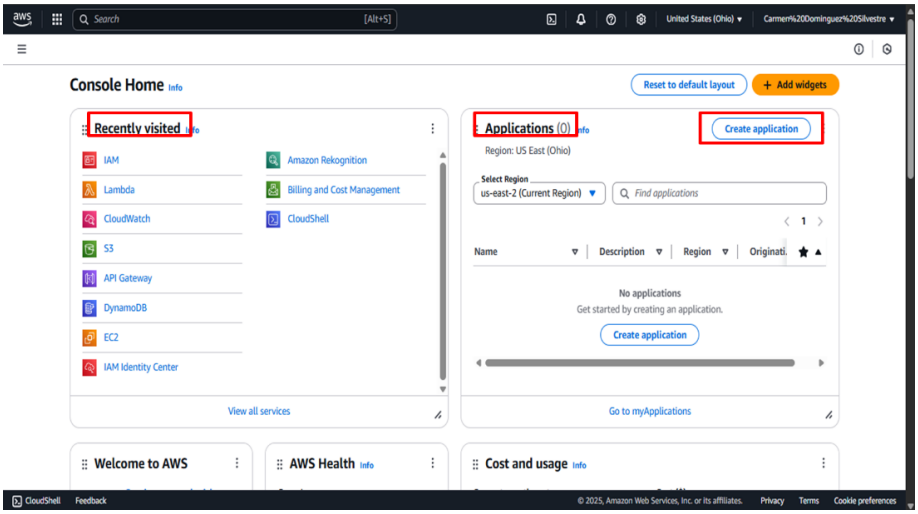

Figura 22. Consola administrativa de Amazon Web Services

3.3.3. Crear un Buckets en Amazon S3

Se crea un buckets en el servicio Amazon S3, que permite almacenar diferentes archivos, este servicio es uno de los más importante. Hacer clic en **"Create Bucket"** (ver figura 23). A continuación, se desplegará un apartado donde se encuentra parámetros importantes para crear el bucket.

Figura 23. Crear bucket en Amazon S3

En el apartado se encuentra los siguientes parámetros: General configuration, Object Ownership, Block Public Access settings for this bucket, Bucket Versioning, Tags, Default encryption y Advanced setting.

En la sección **General configuration** son parámetros básicos para crear el bucket. La región que se selecciono es **US East (Ohio) us-east-2**. El tipo de bucket elegido es **General purpose** se recomienda en la mayoría de los casos ya que va a permitir almacenar los datos en diferentes zonas. Por último, el nombre asignado al bucket es **"prueba-tesis-1"** (ver figura 24).

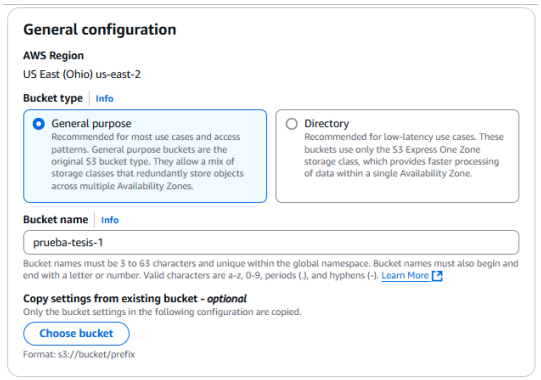

Figura 24. Configuración del bucket – General Configuration

En la sección **Object Ownership** se configura la propiedad del objeto almacenado en el bucket. Es seleccionado la opción **"ACLs disabled (recommended)"**, se refiere a que todos los objetos del bucket serán propiedad de la cuenta de AWS. Por medio de políticas se controlará el acceso y no por control de acceso (ver figura 25).

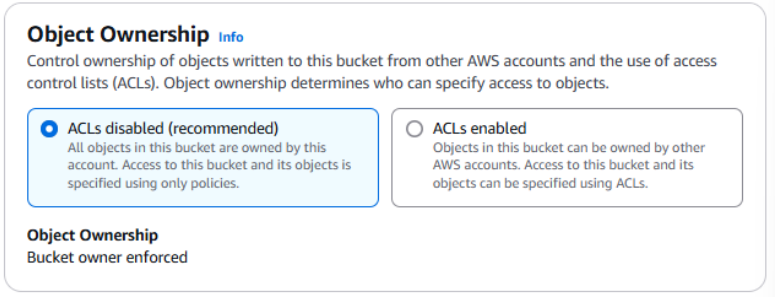

Figura 25. Configuración del bucket - Object Ownership

En la sección **Block Public Access setting for this bucket** es para bloquear el acceso al público en los objetos ya agregado en el bucket (ver figura 26)**.**

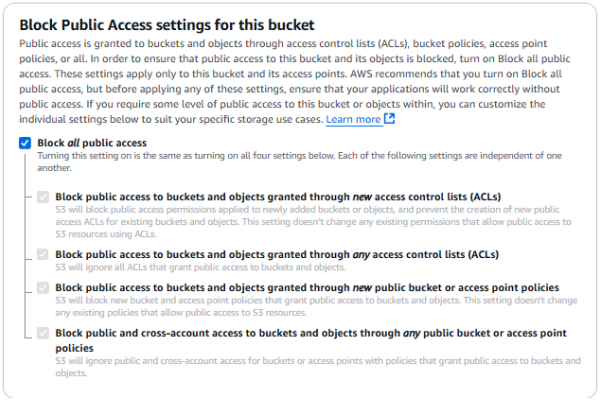

Figura 26. Configuración del bucket - Block Public Access setting for this bucket

En la sección **Bucket Versioning** se selecciona la opción **enable** para mantener múltiples objetos en el mismo bucket y activa el versionada (ver figura 27).

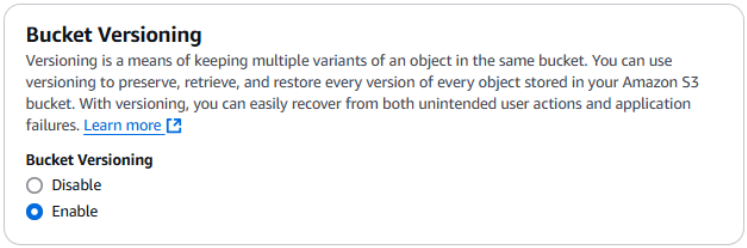

Figura 27. Configuración del bucket - Bucket Versioning

En la sección **Tags** se puede utilizar para mejorar la administración, el control financiero y la estructura jerárquica de todos los recursos de la nube (ver figura 28).

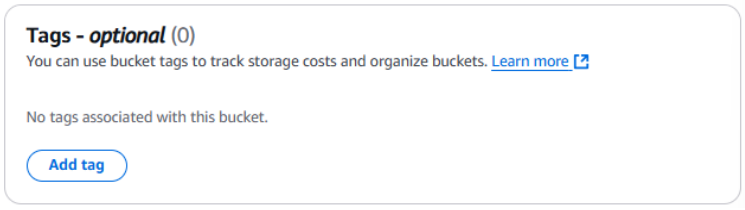

Figura 28. Configuración del bucket - Tags

En la sección **Encryption Type** se seleccionó la opción más básica de AWS que es el **cifrado con claves gestionadas por S3 (SSE-S3)** esto permite proteger los datos contra acceso

no autorizados, la función **Bucket Key** reduce los costos de cifrado con SSE-KMS se escoge la opción **enable** es la más recomendable para ahorrar costos (ver figura 29).

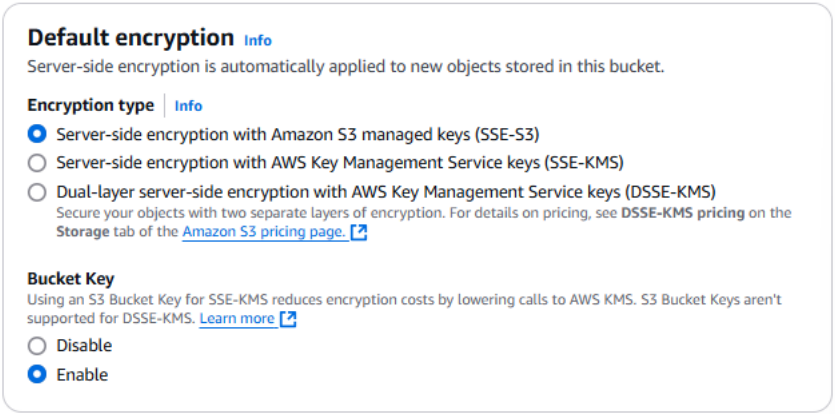

Figura 29. Configuración del bucket - Default encryption

En la sección **Advanced setting** es una configuración avanzada en la que **Object Lock** es una herramienta importante para todos los entornos que demandan inmutabilidad y consta con cumplimiento estricto. Para este caso se selecciona **disable** ya que no se demanda inmutabilidad. Por último, hacer clic en **"Create bucket"** (ver figura 30).

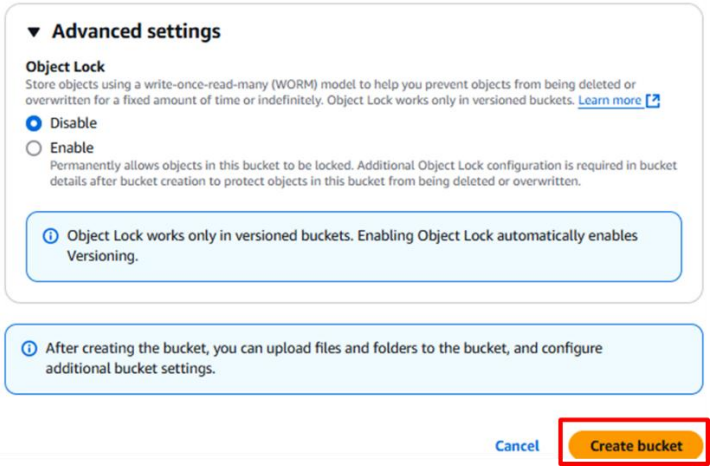

Figura 30. Configuración del bucket - Advanced settings

3.3.3.1. Subir los archivos en el bucket

Luego que el bucket es creado aparece una ventana que es la principal del bucket en donde se sube los objetos en este caso serán las fotos que estarán en la base de datos de AWS. Para subir las fotos hacer clic en **"Upload"** (ver figura 31).

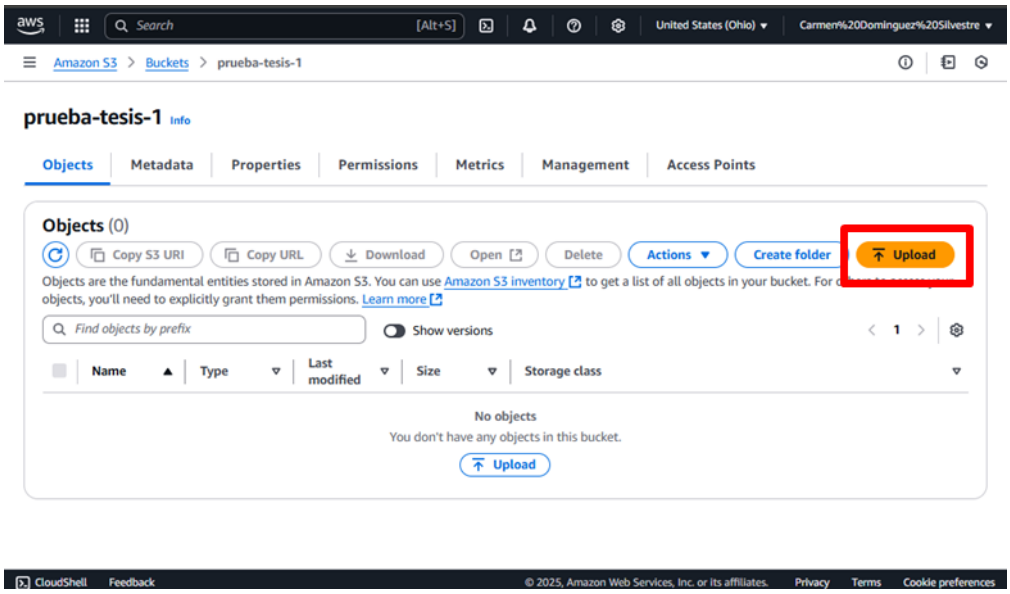

Figura 31. Ventana principal del bucket creado

Hacer clic en **"Add files"** luego se dirige al explorador de archivos en donde se va a seleccionar las fotos que se va a subir a la nube, cuando los archivos este cargando en files and folders hacer clic en **"upload"** (ver la figura 32) como las imágenes están cargadas correctamente.

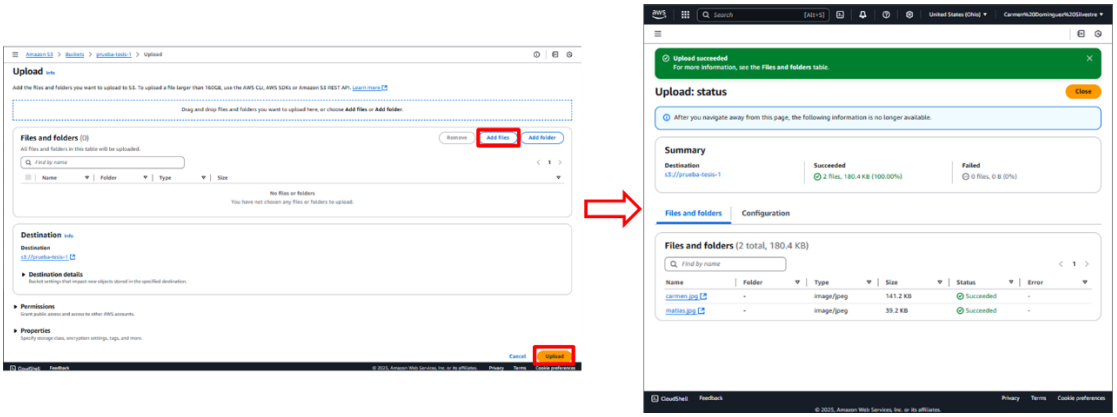

Figura 32. Subir las fotos que estarán en la DB al bucket

3.3.3.2. Almacenar las fotos en la base de datos de Amazon Web Services (AWS)

Para el siguiente paso tener previamente instalado la última version de AWS CLI que se encuentra en el siguiente enlace **https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html**. Abrir el CMD del computador para hacer la respectiva configuración y el comando a utilizar es "aws configure" en donde se colocará el AWS Access Key Id, AWS Secret Access Key, Default región name (allí se coloca la región con la que está configurado AWS) y default output format (el formato de salida) (ver figura 33).

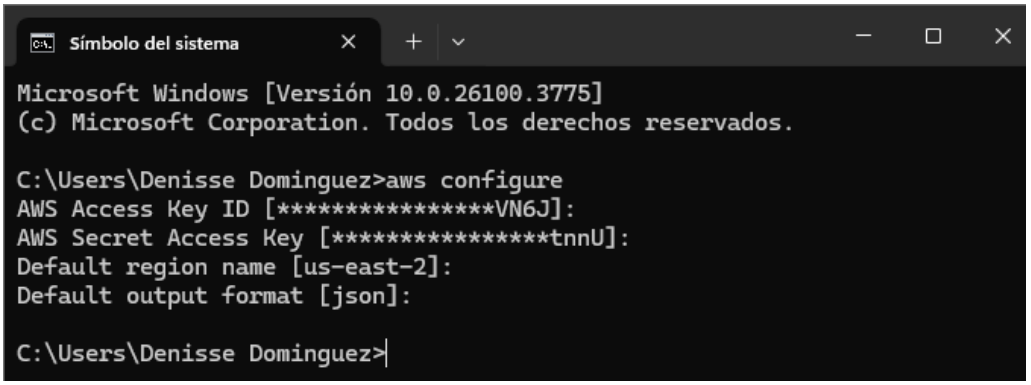

Figura 33. Configurar el CLI con las credenciales del usuario

A continuación, se crea una colección con nombre "Colección_10" en donde se almacenarán las fotos con el siguiente comando **aws rekognition create-collection --collection-id "Coleccion_1". Después asignar las imágenes a la colección con el siguiente comando**

aws rekognition index-faces –image "{\"S3Object\":{\"Bucket\":\"prueba-tesis-1\",\"Name\":\"matias.jpg\"}}" --collection-id "Coleccion_10" --max-faces 1 --quality-filter "AUTO" --detection-attributes "ALL" --external-image-id "Matias este proceso se hace con todas las fotos que serán subidas en bucket (ver figura 34).

```
C:\Users\Denisse Dominguez>aws rekognition create-collection --collectio
n-id "Coleccion_10"
{
    "StatusCode": 200,
    "CollectionArn": "aws:rekognition:us-east-2:820242919138:collection/
Coleccion_10",
    "FaceModelVersion": "7.0"
}


C:\Users\Denisse Dominguez>aws rekognition index-faces -image "{\"S3Obje
ct\":{\"Bucket\":\"prueba-tesis-1\",\"Name\":\"matias.jpg\"}}" --collect
ion-id "Coleccion_10" --max-faces 1 --quality-filter "AUTO" --detection-
attributes "ALL" --external-image-id "Matias"
```

Figura 34. Crear colección y subir las fotografías a Amazon Rekognition

Después se puede observar el análisis de la foto subida en la que es identificado como “Matias” en la que se detectó un rostro masculino, entre 2 a 6 años, sin barba ni bigotes y una expresión mayormente confundida. Se analiza que la imagen tiene buena calidad y fue reconocida con un 99.99% (ver figura 35).

Figura 35. Análisis de la fotografía subida a Amazon Rekognition

3.3.4. Crear Buckets para almacenar las capturas

Se crea un nuevo buckets con nombre **“foto-capturas”** es allí donde se almacenará todas las imágenes capturadas durante el proceso de validación (ver figura 36).

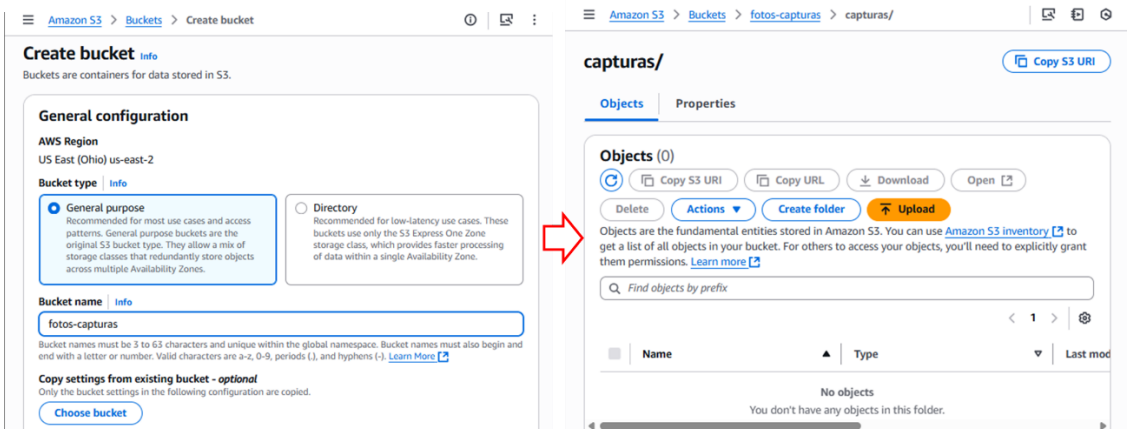

Figura 36. Buckets para almacenar las imágenes

Agregar la política que permita tener acceso al buckets de almacenamiento (ver figura 37). En el Anexo 1 se encuentra el código completo.

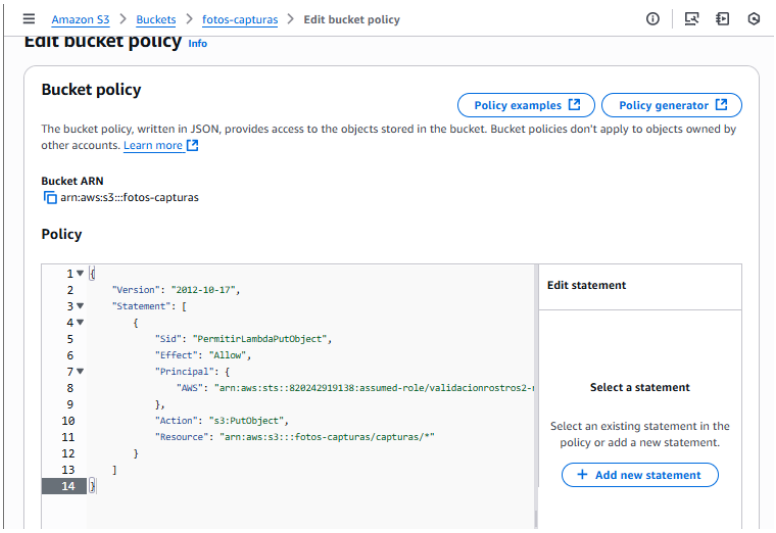

Figura 37. Política del buckets de almacenamiento

3.3.5. Crear la base de datos en DynamoDB

Para crear una base de datos en AWS, se utiliza DynamoDB, es uno de los servicios que facilita este proceso en la que se almacena información de las personas, hacer clic en **"Create table"** (ver figura 38).

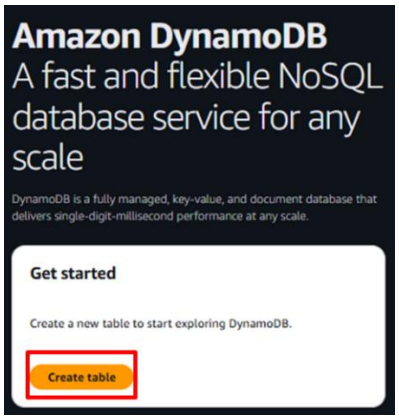

Figura 38. Crear una tabla en DynamoDB

Al hacer clic en **"Create table"**, se despliega una ventana en donde se hace la configuración de la tabla. Ingresa **"DB_Tesis"** en el campo **"Table name"** y establece **"faceId"** como la **"Partition key"**. Este faceId, elegido con precisión, será la clave para identificar a cada persona en la base de datos. Por último, hacer en **"Create table"** y es así como la tabla estará creada (ver figura 39).

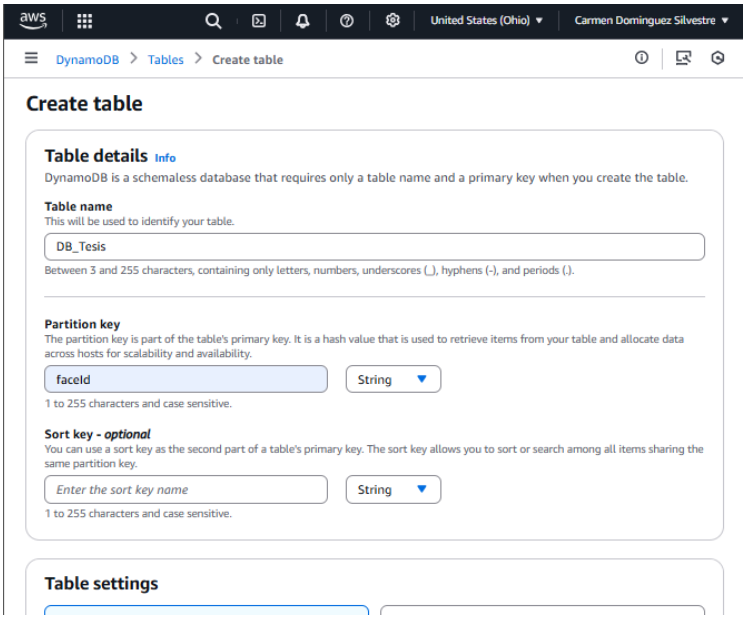

Figura 39. Configuración para crear una tabla en DynamoDB

Una vez creada la tabla, él se dirige nuevamente al servicio de Lambda, con la determinación de configurar un nuevo rol. Navega hacia la sección de configuración, luego a permisos, y accede a la opción de roles. Para configurar el nuevo rol, hace clic en **"Create Policy"**. Agrega el nuevo código y hacer clic en **"Next"**, en la que se permite asignar un nombre a la política. Por último, hacer clic en **"Create Policy"** (ver figura 40). En el Anexo 2 se encuentra el código para el permiso de DynamoDB.

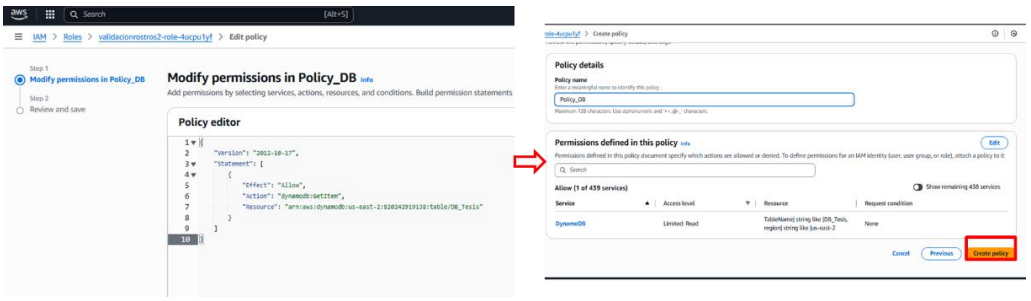

Figura 40. Configuración de permisos para DynamoDB

Con la política ya creada, accede a **"Explore item"**, luego a **"Create ítems"**. Se abre una ventana que pide crear campos a llenar, con los atributos Nombres, Apellidos, Cédula y

cualquier otro campo que desee incluir. Llenar el campo faceId con lo copiado anteriormente. Finalmente, hacer clic en **“Create item”**(ver figura 41).

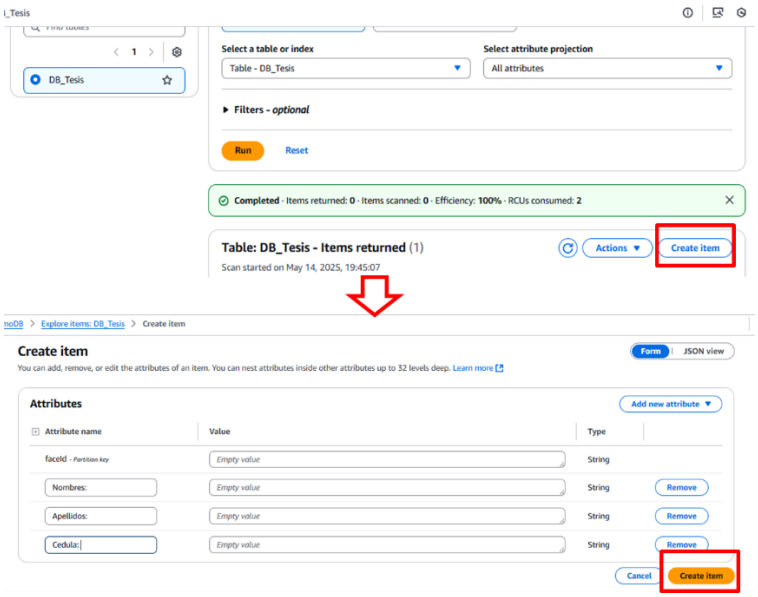

Figura 41. Crear atributos para la tabla

Si se olvida copiar el faceId al momento de subir la imagen mediante código, se creará un código que le permite recuperar el faceId de las fotos ya cargadas en la colección creada en Amazon Rekognition. Es un recurso a utilizar para obtener la información necesaria(ver figura 42).

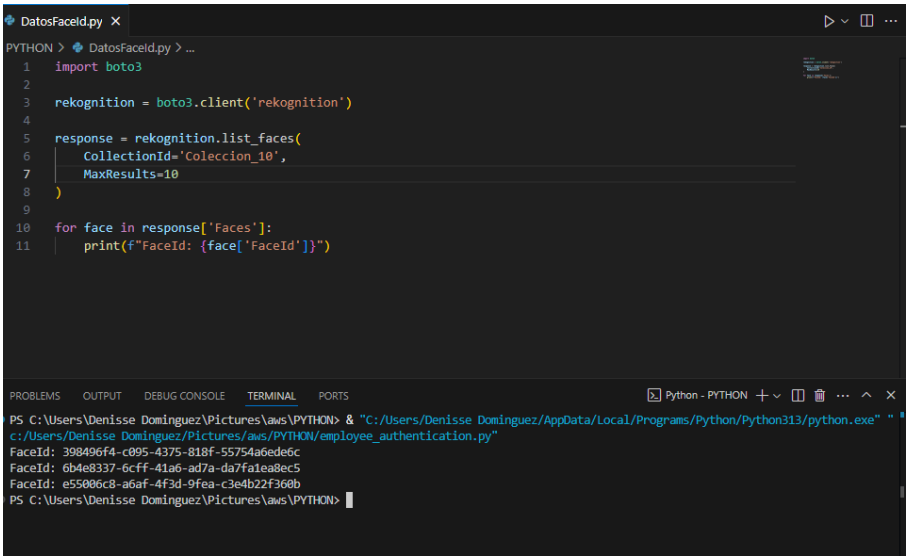

Figura 42. Código para ver el faceId de las fotografías en Amazon Rekognition

Para terminar de crear la tabla de la base de datos, se observan todos los detalles de la tabla y de los ítems creado con la información que se ha añadido. Si se desea incorporar la información de más personas, se deben repetir los pasos descritos anteriormente. De esta manera, se obtiene una base de datos lista para ser utilizada(ver figura 43).

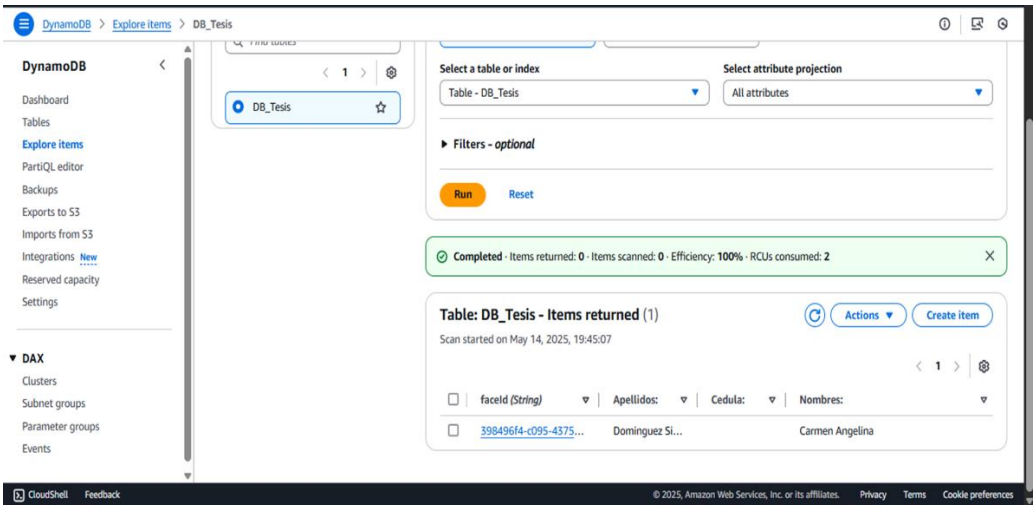

Figura 43. Ítems ya creados con la información de la persona

3.3.6. Crear un cliente MQTT en el servicio IoT Core4

Para realizar la comunicación mediante el protocolo MQTT en Amazon Web Services, es necesario utilizar el servicio AWS IoT, ya que instalar un cliente MQTT de forma independiente puede generar pequeños problemas. Es más seguro y fiable crear un cliente MQTT en AWS IoT. Para iniciar con la configuración, se debe dirigir al servicio y hacer clic en **"Connect device"**(ver figura 44).

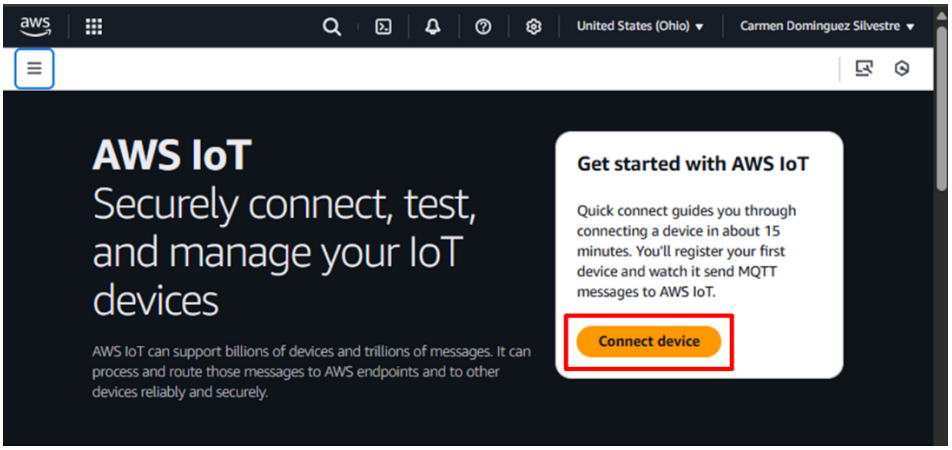

Figura 44. Crear MQTT en IoT Core

Se despliega una ventana donde inicia la configuración. En la primera parte, se observa un ping (endpoint). El endpoint visible se copia, ya que será necesario más adelante, y luego se hace clic en **"Next"**. En la siguiente sección, se observa "Thing properties", donde se crea un nuevo thing con el nombre **"ValidationDevice"**. Así se denominará el thing para ser utilizado posteriormente y poder conectarlo. Para seguir con la configuración se da clic en **"Next"**(ver figura 45).

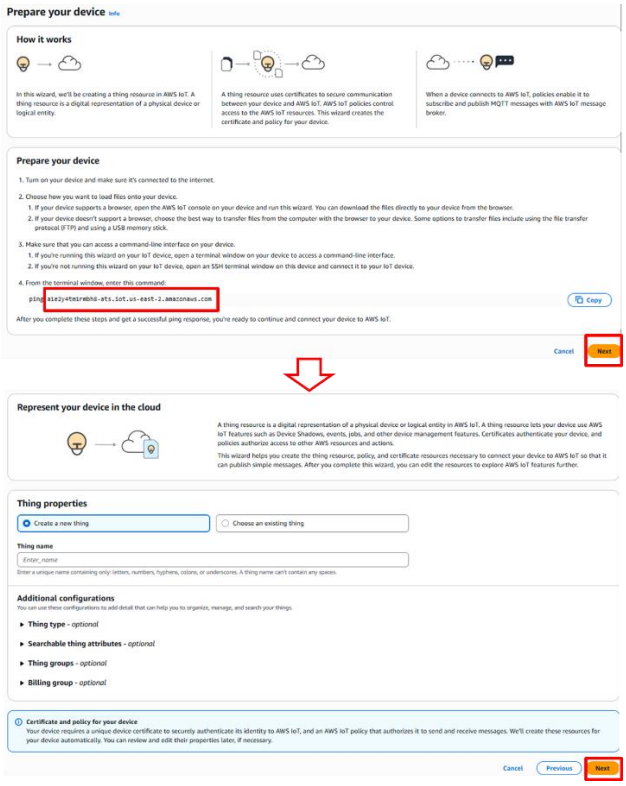

Figura 45. Configuración del cliente de MQTT

En Chosse platform and SDK se selecciona el sistema en el que está operando, en este caso en es **Linux/macOS** y se elige en **Python** el AWS IoT Device SDK, porque es el lenguaje utilizado en el código lambda, hacer clic en **"Next"** para seguir a la siguiente parte de la configuración.

En esta siguiente etapa, se proporciona un kit que incluye **Certificate**, **Private Key**, **Policy**, **Public** y **Script to send and receive messages**, los cuales son necesarios para conectar el cliente. Antes de hacer clic en **"Next"**, no olvidar hacer clic en **"Download connection kit"**(ver figura 46).

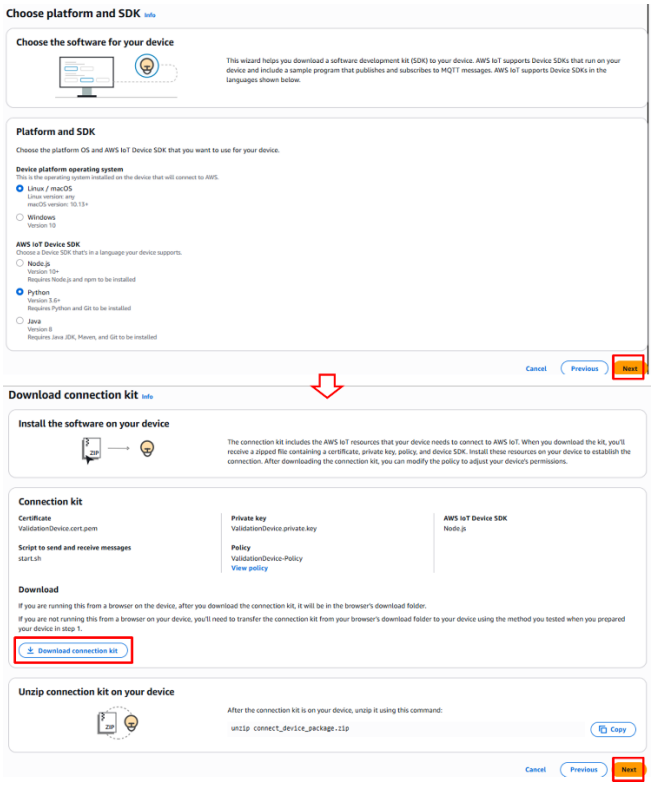

Figura 46. Descargar los archivos para conectar el cliente

Estos son los archivos que se deben descargar en el escritorio y necesario para conectar al servidor (ver figura 47).

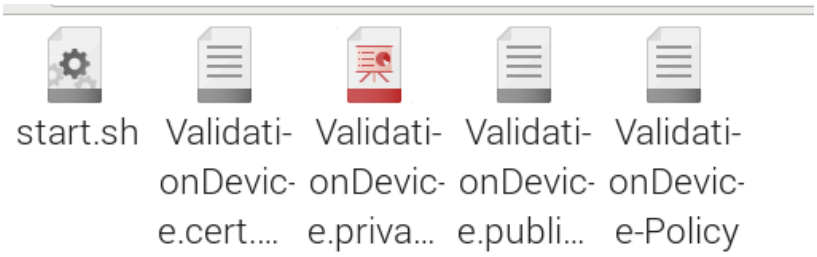

Figura 47. Archivos descargados para conectar al cliente

Descargar el archivo **"AmazonRootCA1.pem"** es un certificado de autorización raíz proporcionado por Amazon Trust Services para una conexión segura. Desde el terminal, se descarga el archivo con el script **"wget -P /home/Carmen/iot_connection_kit/ https://www.amazontrust.com/repository/AmazonRootCA1.pem"**. Este archivo se agrega automáticamente a la carpeta proporcionada junto con los archivos anteriores (ver figura 48).

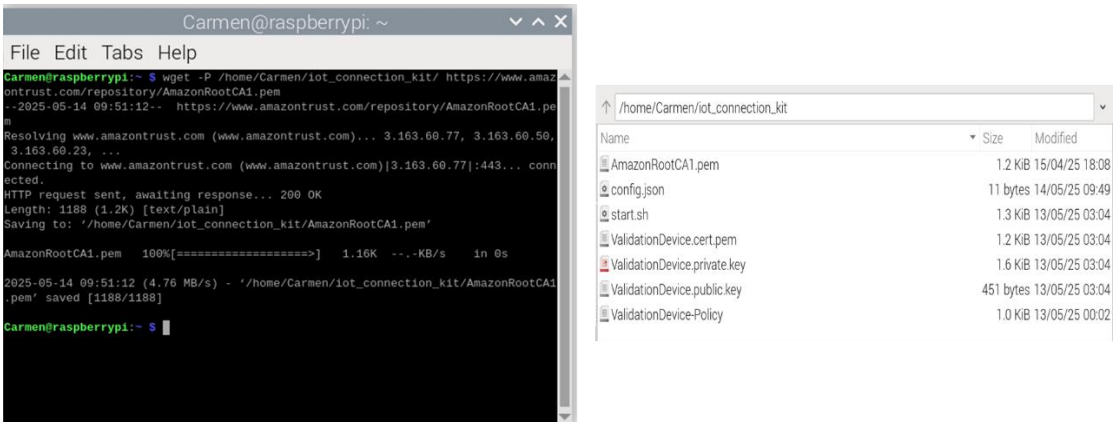

Figura 48. Descargar el archivo AmazonRootCA1

Para realizar la respectiva prueba y verificar si se conecta al Raspberry, se crea una carpeta utilizando el comando **"mkdir iot_connection_kit"**. Este comando generará dicha carpeta. Siguiendo, se crea un archivo .json mediante el comando **"nano"** seguido de la ruta de la carpeta. Luego, presiona **"Enter"**, lo que abre una ventana donde se puede escribir el código (ver figura 49).

Figura 49. Crear un archivo .json para conectar el cliente

Como se mencionó anteriormente, en esta ventana se puede escribir el código. En el archivo .json, se escribe el código para conectar al cliente (ver figura 50). En el campo **"endpoint"**, se coloca el endpoint copiado anteriormente. En **"cert"**, **"key"** y **"ca"**, se indica la ruta en donde se encuentran estos documentos. En **"thingName"**, se especifica el nombre con el que se creó el Thing previamente, y se define el puerto en el que se va a trabajar.

Figura 50. Código para conectarse con el cliente

Para ejecutar y verificar, se crea un archivo .py con el comando **"nano"**, genera un archivo y escribe el código para hacer la respectiva prueba en IoT Core. Es importante verificar dónde se crea el archivo .json, ya que será leído por el archivo .py (ver figura 51). Si el archivo se guardó en otra ruta, se debe especificarla en **"with open"**. Para finalizar, se guarda el archivo con Ctrl+O, Enter y Ctrl+X.

Para ejecutar en el terminal, se abre cada carpeta con **"cd"** hasta llegar a la carpeta donde está el archivo. Una vez allí, con el comando **"python iot_connect.py"** puede ejecutar dicho código.

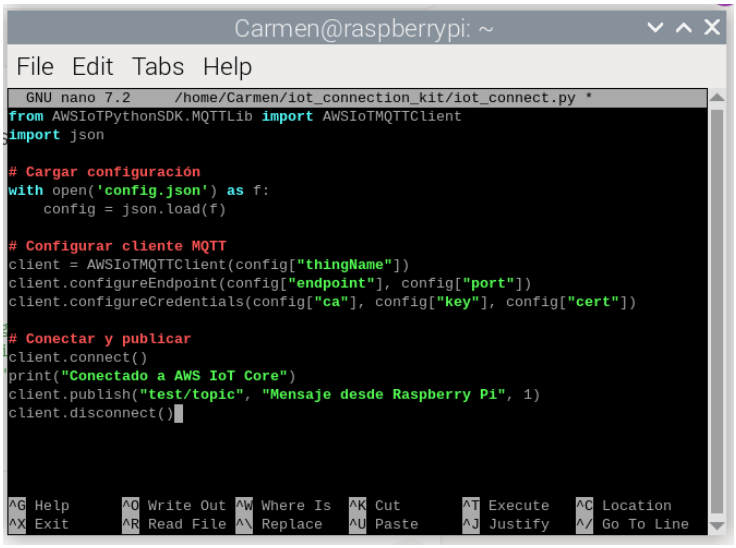

Figura 51. Código para realizar una prueba en el terminal de AWS IoT Core

Después de ejecutar el código, se procede a ir en **“MQTT test Client”**, donde se observa que está conectado. Se realiza un test en **“Topic filter”**, colocando **“test/topic”** y haciendo clic en **“Subscribe”**(ver figura 52).

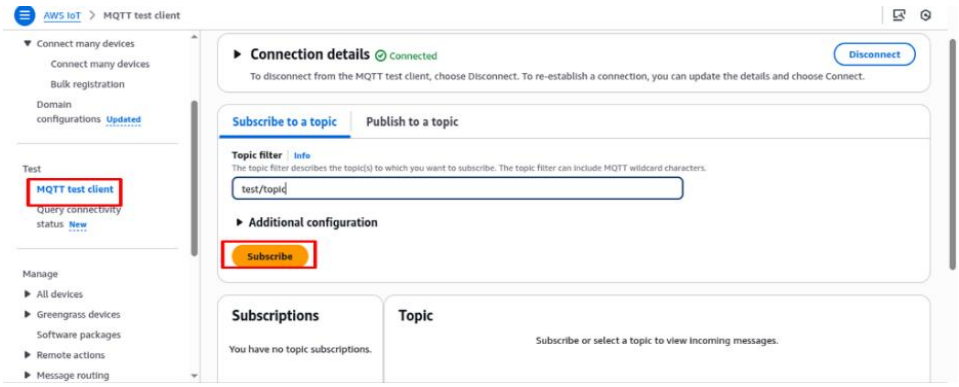

Figura 52 . Hacer test en MQTT Client

En el código creado en la figura 51, la respuesta publicada debería ser **“Mensaje desde Raspberry Pi”**. Sin embargo, se observa que se obtiene una respuesta **“Hello from AWS IoT console”**, lo que indica que se están presentando problemas al momento de publicar (ver figura 53).

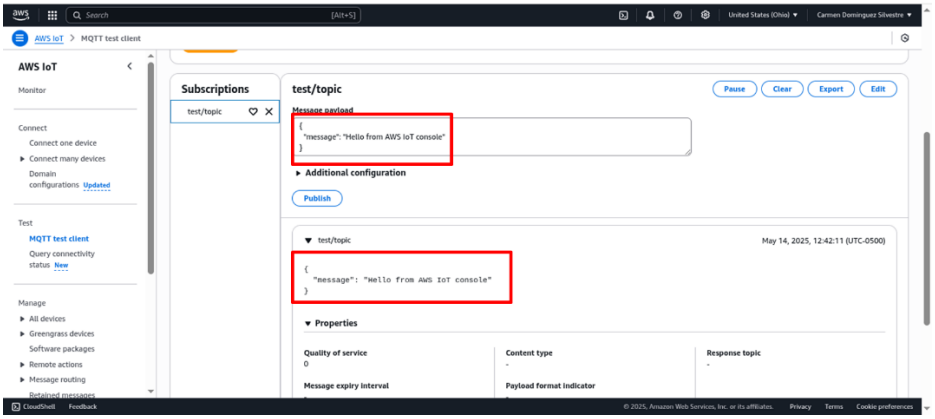

Figura 53. Respuesta del test realizado en MQTT Client

Para el problema presentado crear una política para IoT Core en la que se pueda conectar con el topic de Node-RED, se dirige a **“Security”**, luego a **“Policies”**, y se accede a

“ValidationDevice-Policy”. Se hace clic en **“Edit active version”** y no se debe olvidar dar clic en **JSON** para editar el código en formato JSON (ver figura 54).

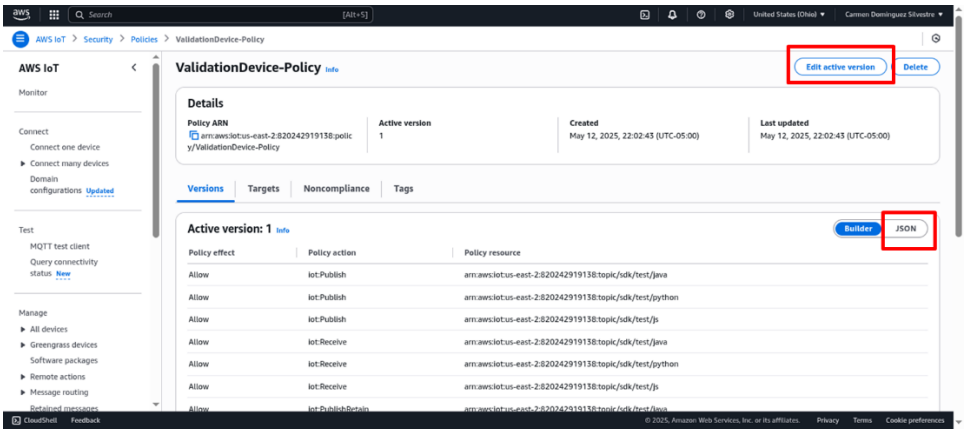

Figura 54. Políticas de ValidationDevice

Se presenta una ventana en donde se editará el código. A continuación, se observa el código que se reemplazará, tomando en cuenta que en **“Resource”** cambiará con el ID de la cuenta de AWS de cada usuario. Al final hacer clic en **“Save as new version”**(ver figura 55). En el Anexo 3 se encuentra el código de la política.

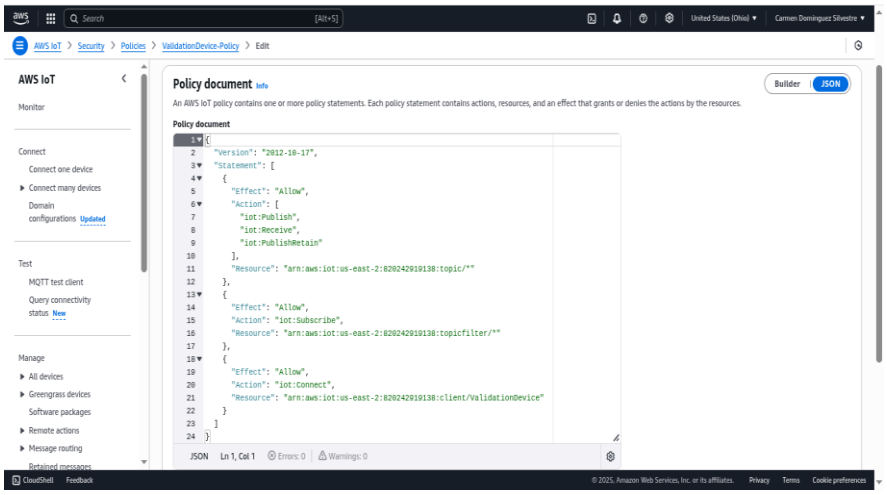

Figura 55. Documento de política corregido para IoT Core

Para que la política se active, se selecciona la segunda política creada después hacer clic en **“Set as active”** y es así como se activa la segunda política(ver figura 56).

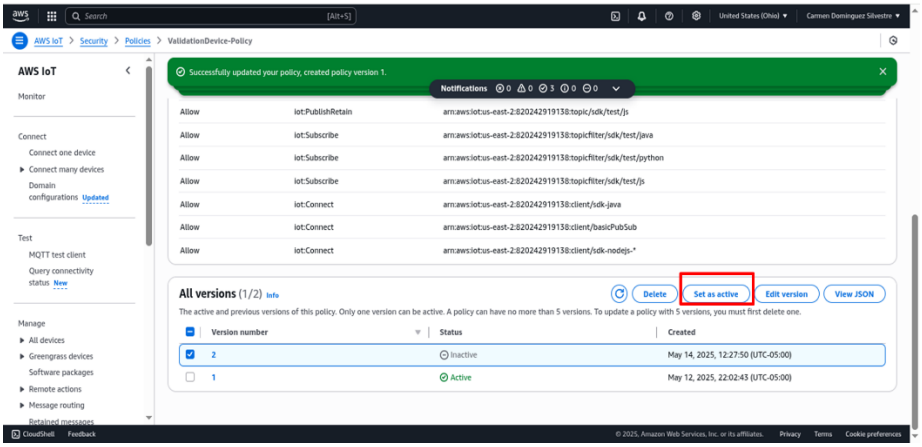

Figura 56. Activar la política creada

Se repite el paso anterior para realizar el respectivo test. Se observa en el terminal que se obtiene una respuesta **"Mensaje desde Raspberry"**, lo que indica que el protocolo MQTT está conectado correctamente y trabajando bien(ver figura 57).

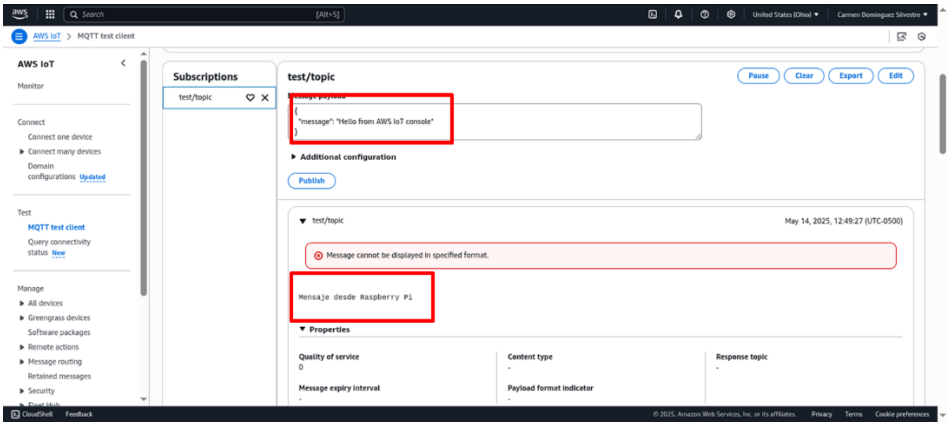

Figura 57. Respuesta del test realizado en MQTT Client

3.3.7. Crear una función en Amazon Lambda

Después de subir las fotos a la nube de AWS y hacer el respectivo análisis de cada una de ellas se crea una función en las se dirige a Amazon Lambda y damos clic en **"Create function"** (ver figura 58).

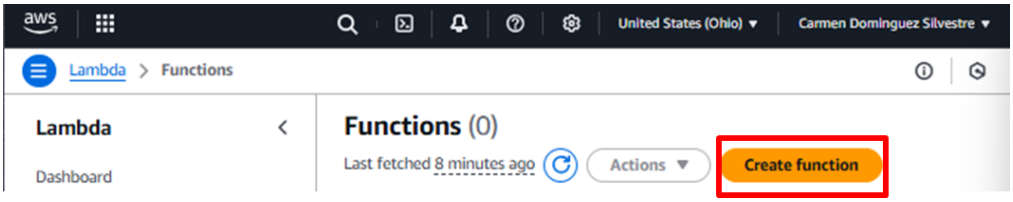

Figura 58. Crear una función en Lambda

En la creación de la función en AWS Lambda, proporcionar opciones como **Author from scratch** (crea una función desde cero), **User a blueprint** (se usa plantillas con código definidos) y **Container image** (usa funciones usando imágenes de docker). En este caso, se usará **Author from scratch** ya que se crea una función desde cero(a).

En **Basic informaction** se especificará el nombre de la función en este caso "validationrostros2" **(a)**, después seleccionar el entorno de ejecución en la que se elegirá Node.js 22x **(b)** con una arquitectura de x86_64 **(c)** para finalizar la configuración hacer clic en **"Create function"** y así estará creada la función (ver figura 59).

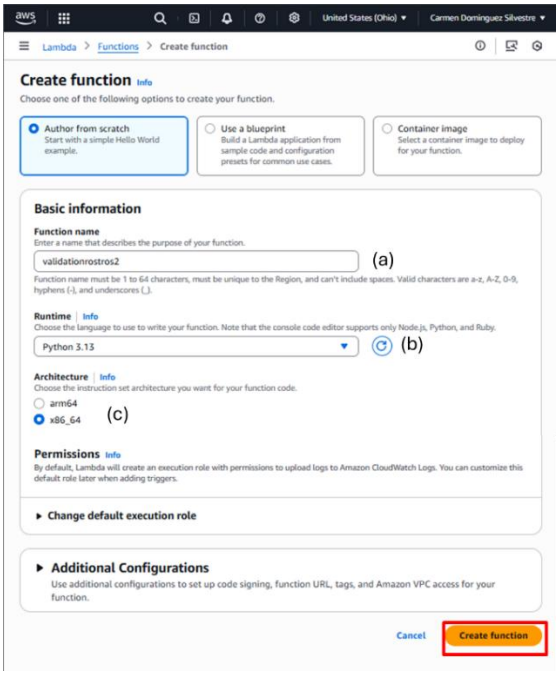

Figura 59. Configurar la función de Lambda

Cuando la función esta creado se tiene una visión general de la función Lambda con la opción de triggers en donde se podrá añadir evento como (API Gateway, S3, etc.), el layers permite incorporar bibliotecas y Destinations es en donde se enviará los resultados a otros servicios. Se detallan la infraestructura en donde se puede ver el ARN de la función que es el identificador único de AWS (ver figura 60).

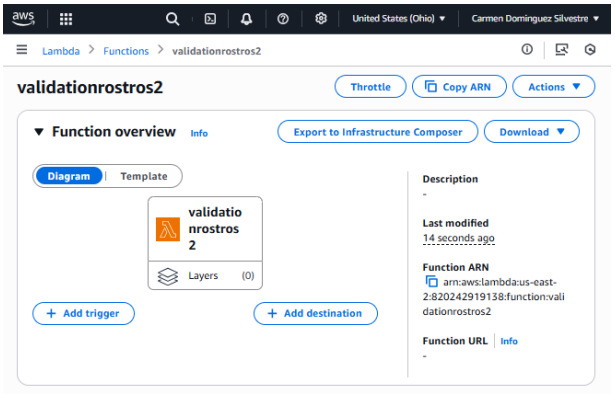

Figura 60. Función de Lambda creada

3.3.7.1. Código fuente en Lambda

En Lambda se observa diferentes opciones como **Code, Test, Monitor, Configuration, Aliases y versions**. En code se va a cargar el código fuente en que la función Lambda en Python recibe una imagen en base64, la procesa con AWS Rekognition para buscar coincidencias faciales (ver figura 61).

Importante colocar el nombre de la colección creada, nombre de la base de datos y del Client MQTT. En el Anexo 4 se encuentra el código completo para lambda.

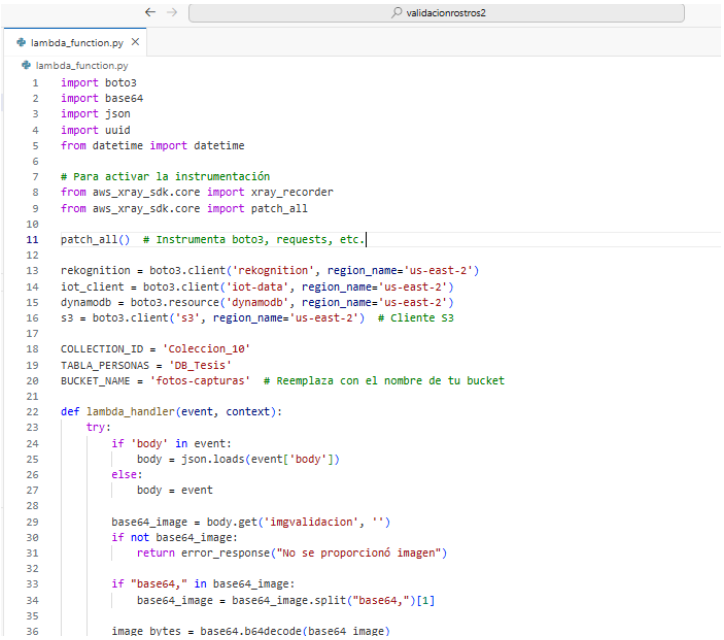

Figura 61. Código de la función Lambda

Luego de colocar el código que se va a entrenar en lambda se dirige a configuration para así dar los permisos y roles en Lambda. Hacer clic en el enlace que se encuentra en **Role name (a)** (ver figura 62).

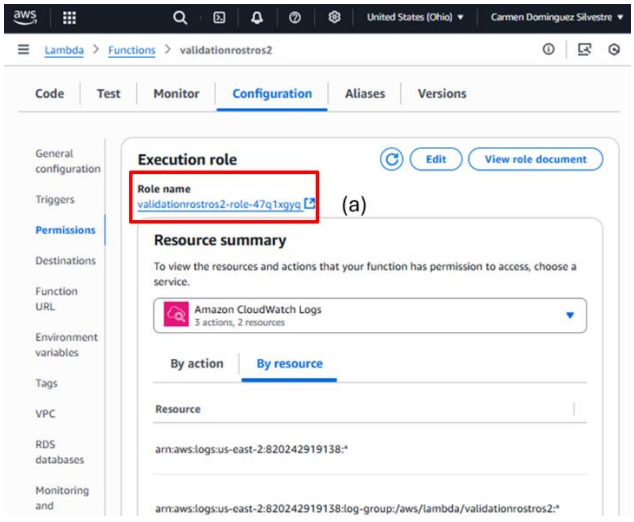

Figura 62. Ejecutar los roles ya creado para la función

En este servicio se encuentra las políticas de permiso, para agregar hacer clic en **"Agregar permisos" (a)**, en la que se seleccionada todas las políticas de permisos ya que son

muy necesarias para acceder a la lambda. Se puede observar todas las políticas que se deben de agregar **(b)** (ver figura 63).

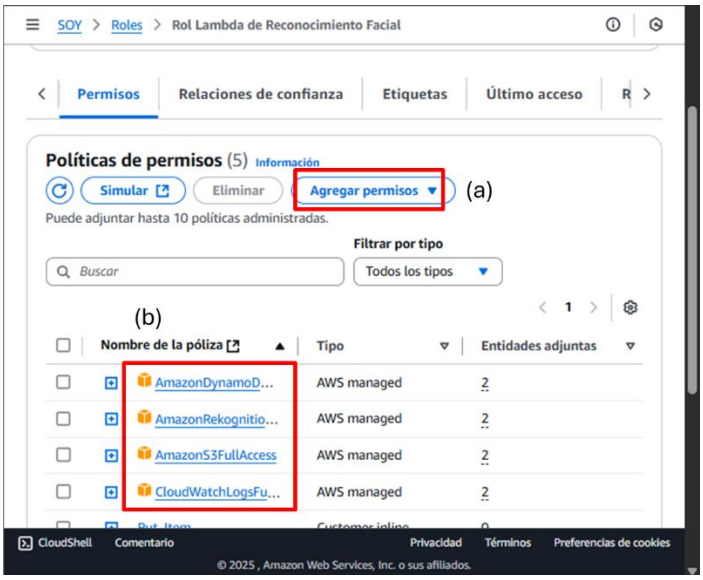

Figura 63. Políticas necesarias para ejecutar la función

3.3.8. Crear API en el API Gateway

Cuando ya se tiene los permisos necesarios se dirige al servicio API Gateway en donde se crea un API que va a permitir conectarse y acceder a datos, el código para el API se ejecuta en AWS Lambda y su comunicación es en tiempo real. Para crear el API hacer clic en **"Create an API"** (ver figura 64).

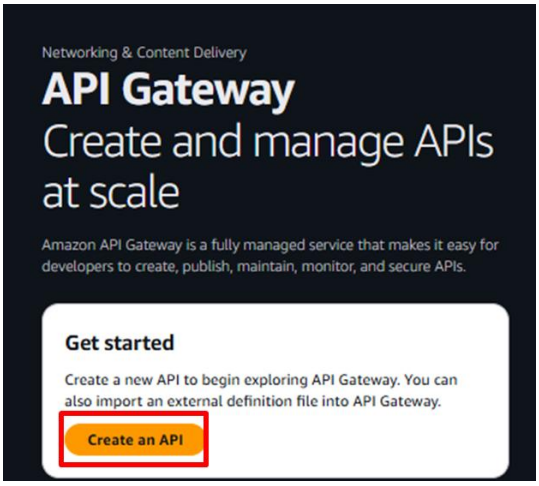

Figura 64. Crear un API en API Gateway

Después de hacer clic en Create an API se elige un tipo de API en este caso se selecciona la opción de **"REST API"** va a permitir el desarrollo de la API con control a las solicitudes y respuestas, también compatible con servicios de lambda y HTTP **(a)** por último hacer clic en **"Build"** para dirigirse a los detalles del API **(b)** (ver figura 65) .

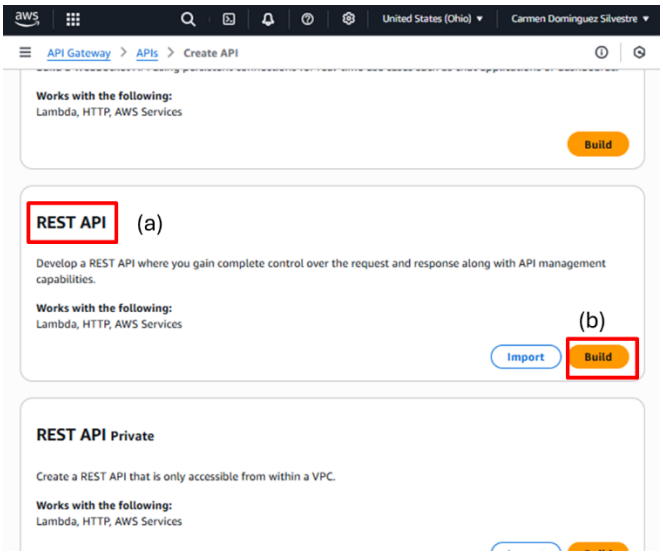

Figura 65. Seleccionar el API correspondiente

Cuando ya se elija el tipo del API se empieza a configurar todos los detalles para la nueva API desde cero con **"New API"**. El nombre que se asignó al API es **"Prueba_tesis"**.

Seleccionar el tipo de endpoint es “Regional” con un soporte para direcciones IP dualstack (IPv4 e IPv6), terminada la configuración hacer clic en **“Create API”** (ver figura 66).

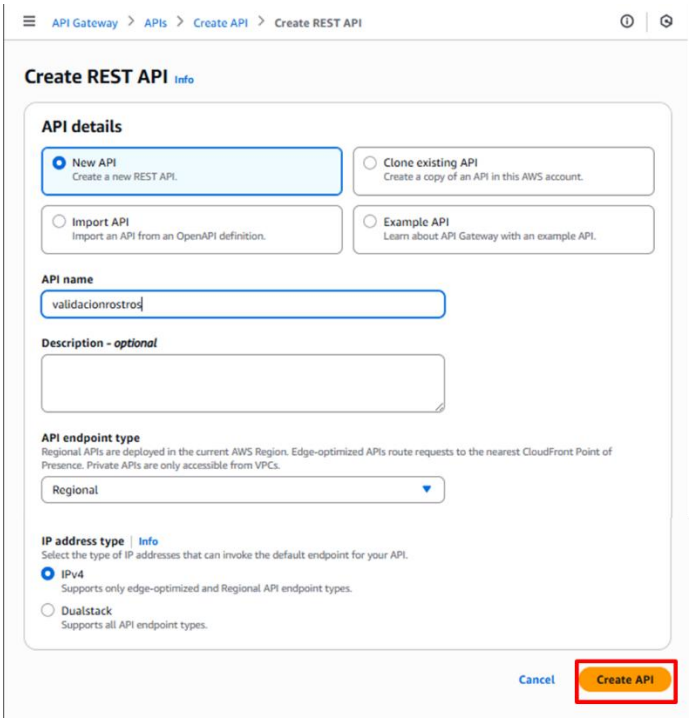

Figura 66. Configurar el API

3.3.8.1. Crear los métodos en recursos

Se observa diferentes secciones en donde recursos es una sección en donde se puede agregar recursos al API, en detalles del recurso se observa el recurso raíz **“/”** el ID del recurso que es un indicador único para el recurso creado. Después de ver todas las secciones del API Gateway hacer clic en **“Create method”** (ver figura 67).

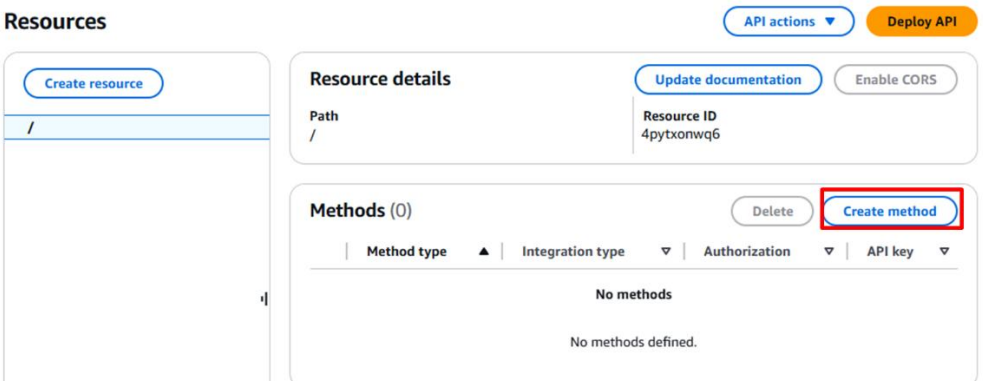

Figura 67. Crear un método en el API

Muestran los detalles del método para configurarla con un método **"POST"** en el API Gateway destacando así que tipo de integración se va a seleccionar "Lambda function", en donde se especifica el ARN de la función Lambda, cuando se termina la configuración hacer clic en **"Create method"** (ver figura 68).

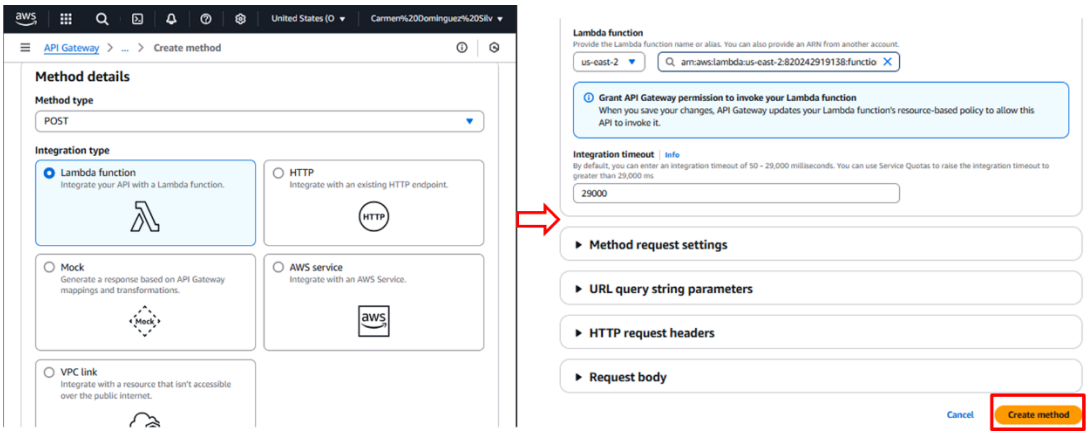

Figura 68. Configurar el tipo de método a utilizar

3.3.8.2. Habilitar CORS

Después de crear el método **"POST"** se observa en la primera imagen un diagrama secuencial en donde se tiene **Client, Method Request, Integration Request, Lambda Integration, Integration response, Method response** y al final su respuesta va al **Client**, para que el método sea habilitado, se configura el **"CORS"** (ver figura 69).

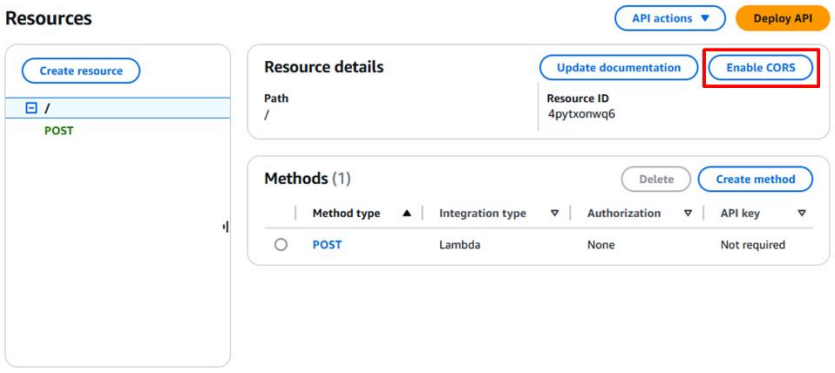

Figura 69. Habilitar los CORS en el API

Para configurar el CORS en Gateway Responses se habilita el método POST en **Access-Control-Allow-Methods**, los valores predeterminados se mantienen en **Allow-Headers** y **Allow-Origin** hacer clic en **Save** para guardar toda la configuración. Cuando se termina se observa un nuevo tipo de método llamado **"OPTIONS"**, por último, hacer clic en **"Deploy API"** para implementar el API (ver figura 70).

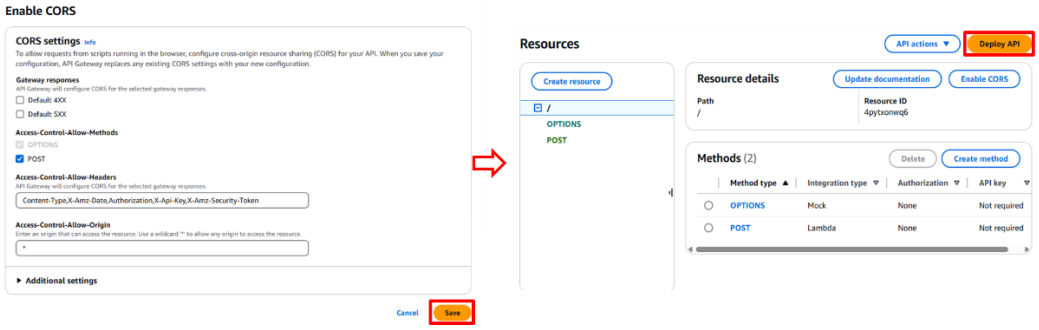

Figura 70. Configurar el CORS del API

3.3.8.3. Crear etapa para el API

Una vez finalizada la configuración del API, para que la URL pública pueda ser utilizada, se debe proceder con el despliegue, mediante la creación de un stage. Al hacer clic en **"Deploy API"**, se despliega una ventana que ofrece la opción de crear un stage. Se selecciona la opción de stage, se elige **"New stage"**, se le asigna el nombre **"test"** y, al concluir, se hace clic en **"Deploy"** (ver figura 71).

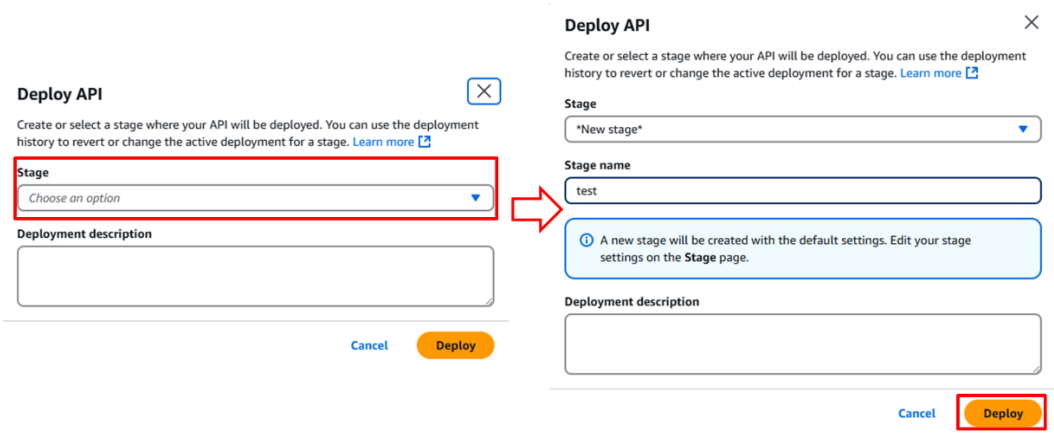

Figura 71. Crear una etapa en el API y desplegarlo

Para completar la configuración, se desplegará una ventana donde el stage proporcionará información sobre la configuración del API, en la que se observa una URL. Esa URL debe ser copiada, para utilizarlo en el código HTML (ver figura 72).

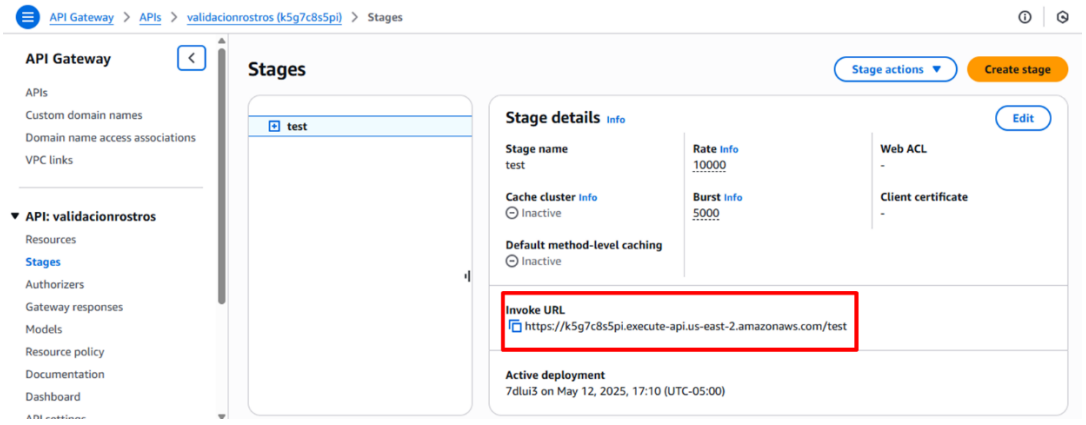

Figura 72. Stage y URL del API creado

3.3.9. Crear código HTML para la interfaz

Para crear un archivo HTML, se creará un archivo .txt que luego podrá cambiar su extensión a .html. En el Bloc de Notas, comenzará a escribir el código con la interfaz adecuada (ver figura 73). Se debe tomar en cuenta que dentro del código habrá una línea llamada **"const apiURL"** en donde se colocará la URL del API que se copió anteriormente. En Anexo 5 se encuentra el código completo de la interfaz de Reconocimiento facial.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sistema de Reconocimiento Facial</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #e6f3fa;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            min-height: 100vh;
            margin: 0;
            padding: 20px;
            box-sizing: border-box;
        }
        h2 {
            color: #1e90ff;
            font-size: 28px;
            margin-bottom: 20px;
            text-align: center;
            text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.1);
        }
        .media-container {
            display: flex;
            gap: 20px;
            align-items: center;
            margin-bottom: 20px;
        }
        #video, #canvas {
            width: 450px;
```

Figura 73. Código de la interfaz en HTML

Cuando se tenga el código completo, se sube al almacenamiento de Amazon S3, se accede al bucket **"prueba-tesis-1"** y se hace clic en **"Upload"** (ver figura 74).

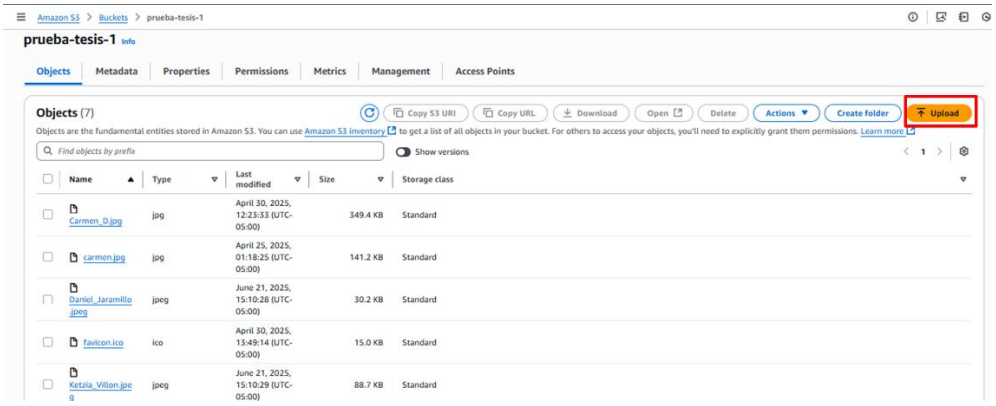

Figura 74. Subir el archivo HTML en Amazon S3

Hacer clic en **"Add files"** para seleccionar el archivo HTML desde el directorio y es así como observa el archivo seleccionado. Después, hacer clic en **"Upload"** para subir el archivo al bucket. Cuando el archivo se ha subido, observa el archivo subrayado, lo que indica que se puede abrir el archivo (ver figura 75).

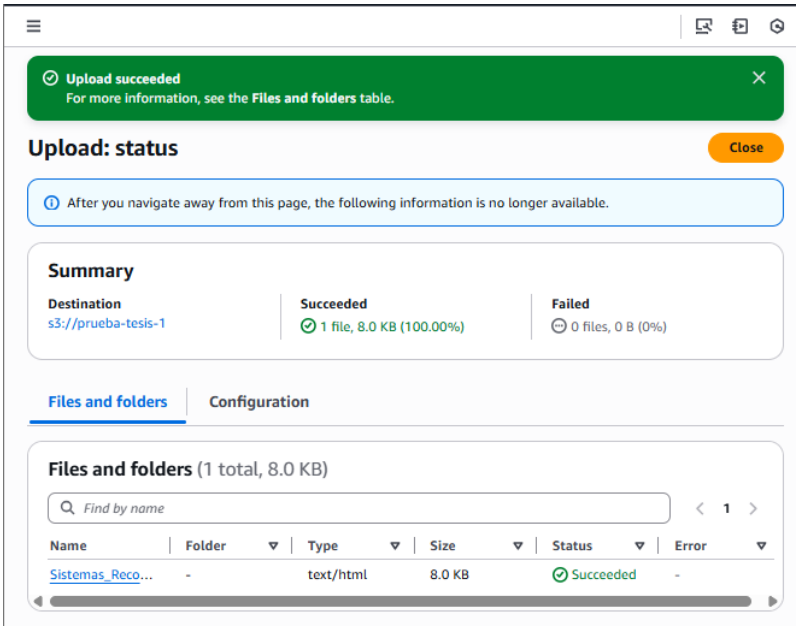

Figura 75. Archivo subido en Amazon S3

Cuando se hace clic en el archivo HTML subido, se despliega una ventana con toda la información del archivo, en la que se genera un URL. Al hacer clic en el enlace, se puede ver la interfaz creada, la que ofrece opciones para capturar y para validar, mostrando los resultados de validación (ver figura 76).

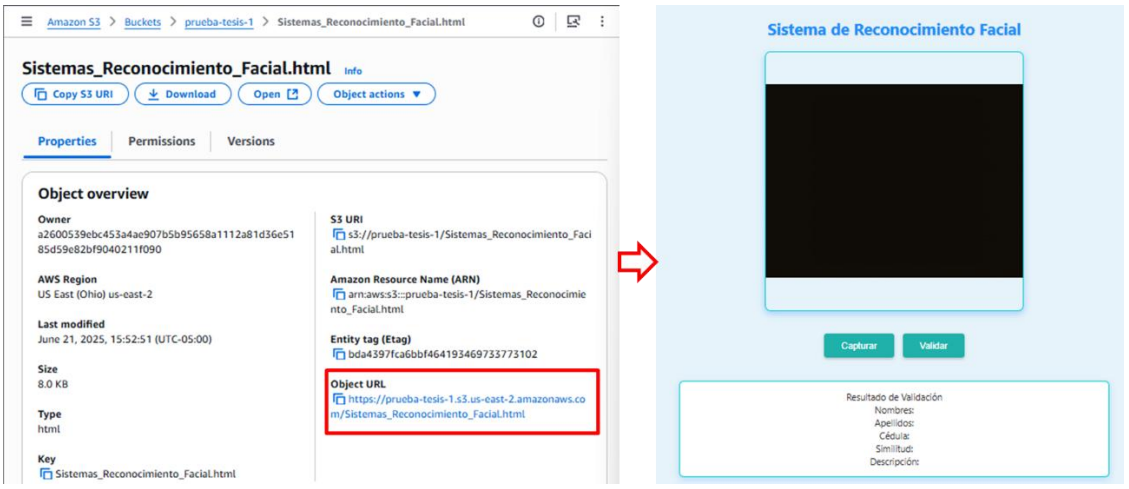

Figura 76. Interfaz de Reconocimiento Facial

3.3.10. Crear código QR para la interfaz y red wifi

Se crea un código QR que facilite el acceso a una interfaz de reconocimiento facial y a una red Wi-Fi, con el objetivo de optimizar la experiencia del usuario. Este código debe codificar un enlace URL que dirija directamente a la página web sin necesidad de ingresar datos manualmente, tanto para el acceso a la interfaz como para la conexión a la red Wi-Fi. La red Wi-Fi se utilizará como alternativa en caso de que el usuario no disponga de acceso a internet (ver figura 77).

Figura 77. Código QR para la red Wi - Fi y a la interfaz de Reconocimiento Facial

3.3.11. Instalacion de Node-RED en el Raspberry Pi

Para instalar Node-RED en el terminal de la Raspberry, se utiliza un script que instala Node.js, Node-RED y configura automáticamente el servicio de Node-RED. El script para emplear es **"bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)".**

Para habilitar Node-RED como servicio, se utiliza el siguiente comando: **"sudo systemctl enable nodered.service"** (ver figura 78).

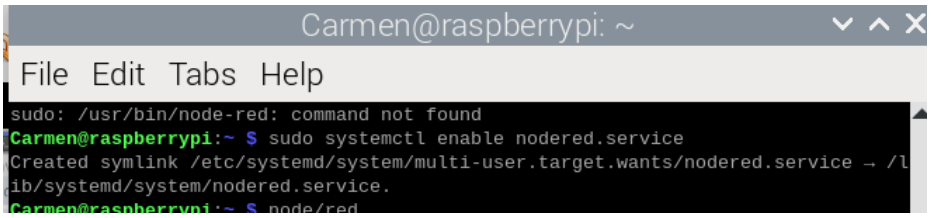

Figura 78. Instalar y habilitar Node-RED

Cuando se habilita Node-RED, con el comando **"node-red"** se inicia un proceso en el que se muestra toda la información durante el arranque. En este proceso, se proporciona un nuevo server, en este caso, **http://127.0.0.1:1880/** (ver figura 79).

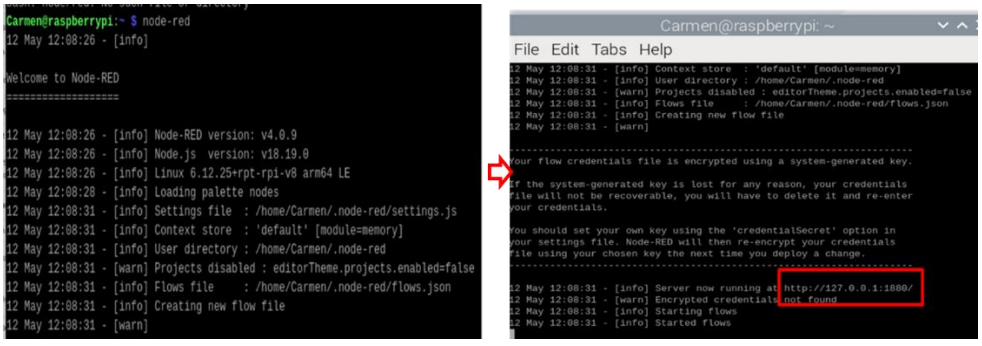

Figura 79. Proporciona la IP para acceder a Node-RED

Se dirige al entorno de node-RED en donde cuentan con muchas funciones de muy importancia (ver figura 80). Es importante instalar nodo GPIO para emplear los pines del Raspberry Pi que se ejecuta con el comando **"npm install node-red-node-pi-gpio"** en el terminal del Raspberry Pi permitiendo así obtener los nodos GPIO.

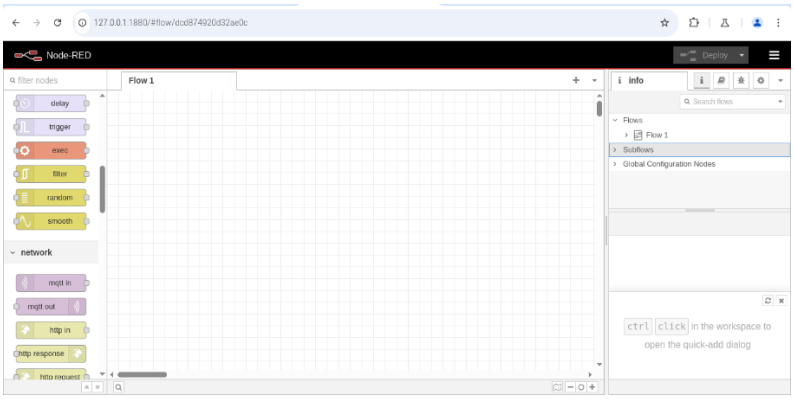

Figura 80. Entorno de Node-RED

3.3.11.1. Crear diagrama de flujo en Node-RED

Se creará un diagrama de flujo funcional para tomar decisiones con los datos recibidos por MQTT desde AWS IoT. El objetivo es establecer una arquitectura hibrida en donde el proceso inicial es en la nube y la respuesta se ejecuta en el Raspberry Pi (ver figura 81).

MQTT In: mediante este nodo se recibirá un mensaje JSON que tendrá un valor de similitud generado desde AWS Rekognition.

Function: el valor que es recibido se convierte en flotante y será evaluado dependiendo de la condición configurada

Rpi-gpio out: actuara como una salida digital para apagar o encender un dispositivo físico.

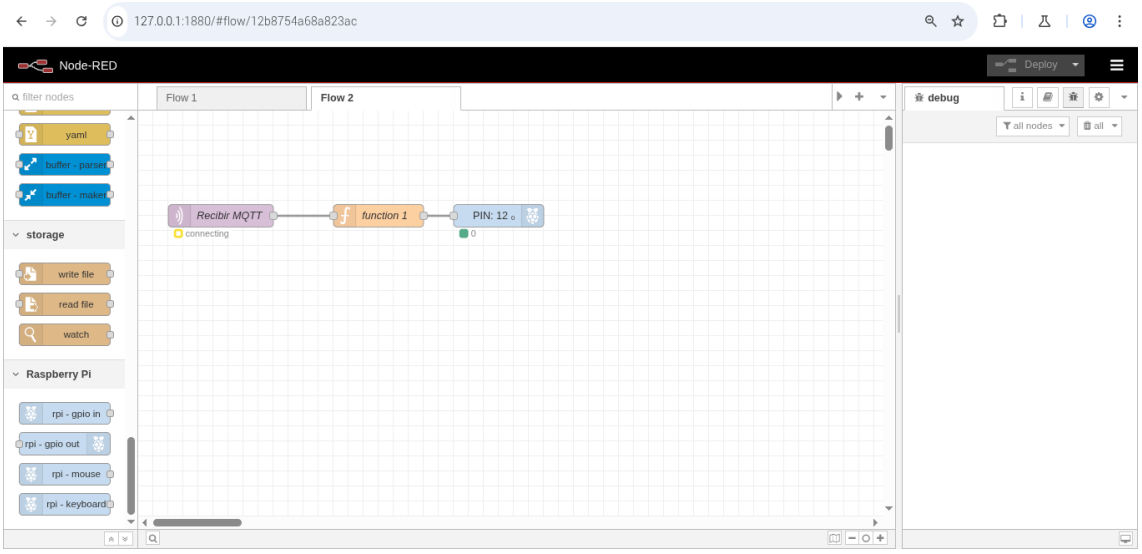

Figura 81. Diagrama de flujo en Node-RED

MQTT input Node

Para recibir mensajes enviados desde el broker MQTT con un topic especifico el nodo será configurado como entrada, se configuran tres parámetros importantes en el nodo, en las que ayudara a conectar con el broker MQTT, los cuales se detallan a continuación:

- Configuración del server
- Configuración TLS
- Configuración general

Configuración del Server

Para configurar el Server se va a llenar los campos con la información para establecer la conexión con el broker de AWS IoT Core. A continuación, se detalla la información a necesaria.

- **Name:** AWS IoT Core
- **Server:** a1e2y4tm1rmbh8-ats.iot.us-east-2.amazonaws.com

- **Port:** 8883
- **Protocol:** MQTT V3.1.1
- **Client ID:** ValidationDevice

Nota: Para conectar MQTT se necesita configurar el TLS, para configurar TLS hacer clic en agregar (+) y cuando se termine la configuración TLS hacer clic en "Update" para subir la información (ver figura 82).

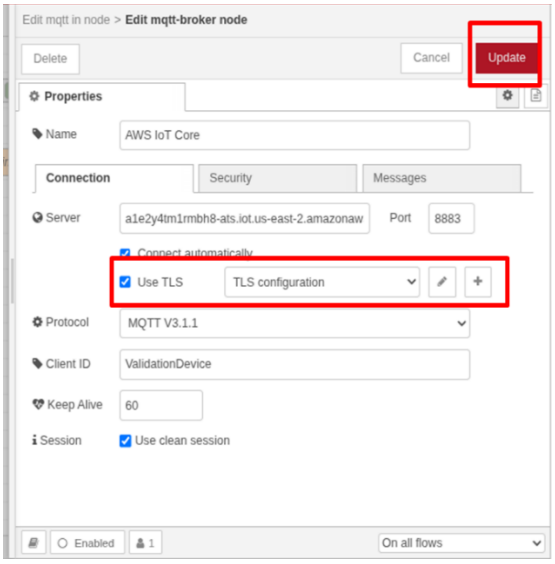

Figura 82. Configuración del Server en el node MQTT

Configuración TLS

En la configuración de TLS se subirá todos los archivos necesarios para la conexión al servidor. Los archivos ya descargados anteriormente. (ver figura 47)

- **Certificate:** ValidationDevice.cert.pem
- **Private Key:** ValidationDevice.private.key
- **CA Certificate:** AmazonRootCA1.pem

Al terminar la configuración TLS hacer clic en **"Update"** (ver figura 83).

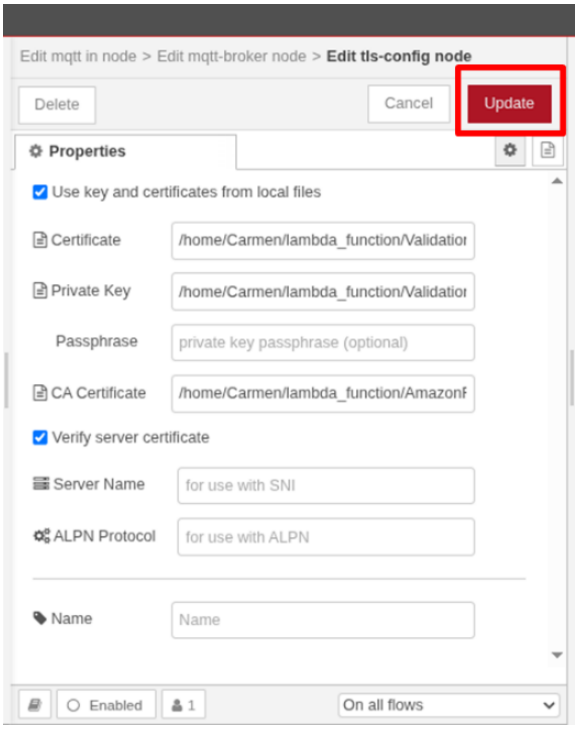

Figura 83. Configuración del TLS en el node de MQTT

Configuración General

Para finalizar con la configuración del nodo, se completan los datos faltantes como es el Topic, QoS, Output y Name, para guardar toda la información subida al nodo hacer clic en **"Done"**(ver figura 84).

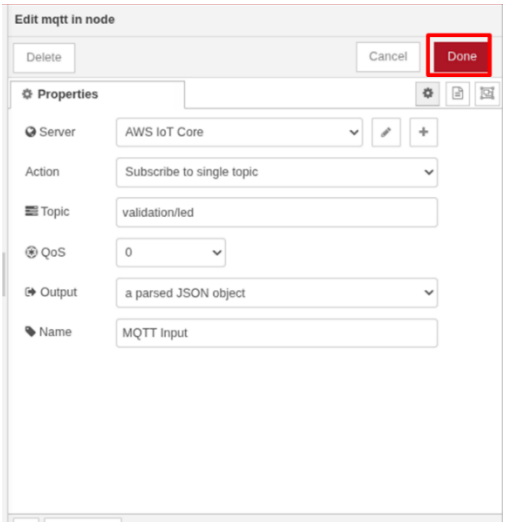

Figura 84. Configuración general en el node MQTT

Function Node

El node va a procesar el mensaje recibido por el protocolo MQTT y comparar, depende del rango que se creó el código ejecutado. En la que si se cumple la condición su resultado será 0 para así activar el pin, en caso contrario va a devolver un 1 (ver figura 85).

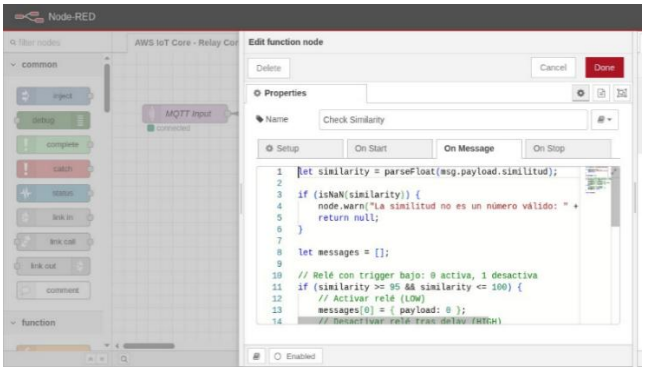

Figura 85. Código para comparar la similitud en la función node

Rpi-gpio out Node

Activar el pin que se va a utilizar y la condición de su salida, activar **"Initialise pin state"** y seleccionar Initial level of pin – low (0) y para guardar la esta configuración hacer clic en **"Done"** (ver la figura 86).

En el Anexo 6 se encuentra el código .json de todo el diagrama de flujo con cada una de sus configuraciones realizadas.

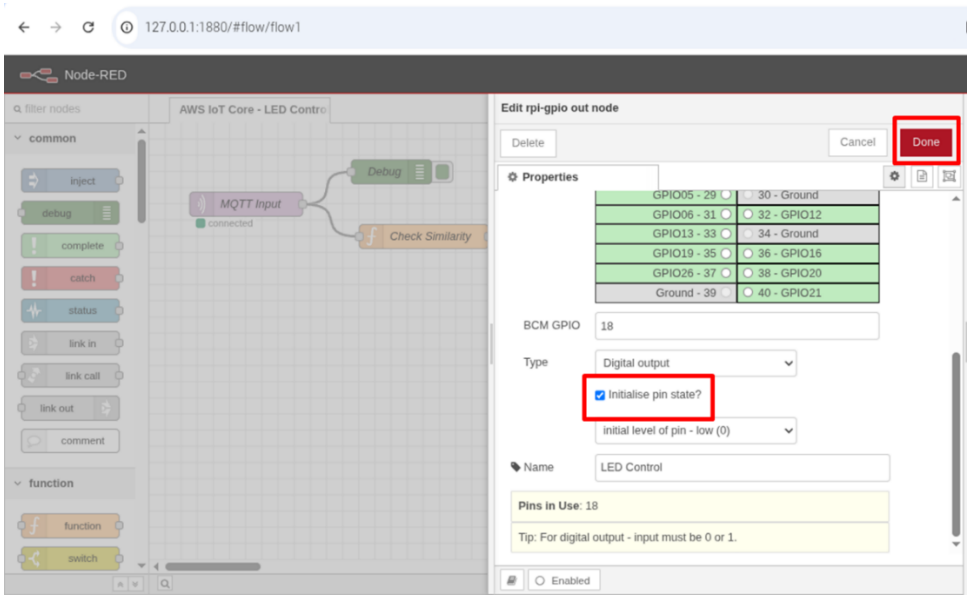

Figura 86. Configuración del node GPIO

3.4. Prueba en el terminal de Node-RED con LED

Se realiza una prueba con LED utilizando como un actuador, en la que se implementó un proceso de reconocimiento facial completo. Desde el interfaz creado se captura la imagen en la que es enviada a Amazon Rekognition para la respectiva validación a través de la función Lambda en AWS. El resultado obtenido al análisis es de 99.99% de similitud, ese resultado es enviado por el protocolo MQTT a través del IoT Core hasta Node-RED, dado que el código fue programado para activar el LED cuando sea mayor a 95%, se procedió a encender el LED y su salida es de 0. En el terminal de Node-RED se mostrará los datos de la persona identificada confirmando que se encuentra en la Base de Datos (ver figura 87).

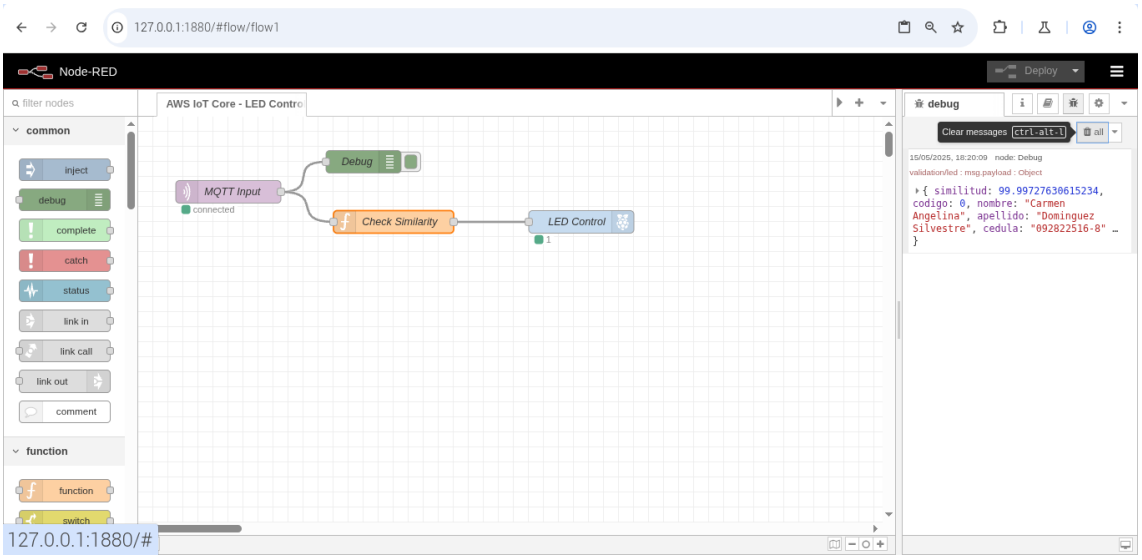

Figura 87. Prueba con una persona que se encuentra en la Base de Datos

En otro caso se probó con otra persona en la que el resultado fue de 0% de similitud significa que no se encuentra en la Base de Datos entonces como el nodo function evalúa la similitud recibida desde Lambda, entonces no se procederá a encender el LED en la que su salida será un 1 y no proporcionará datos en el terminal (ver figura 88).

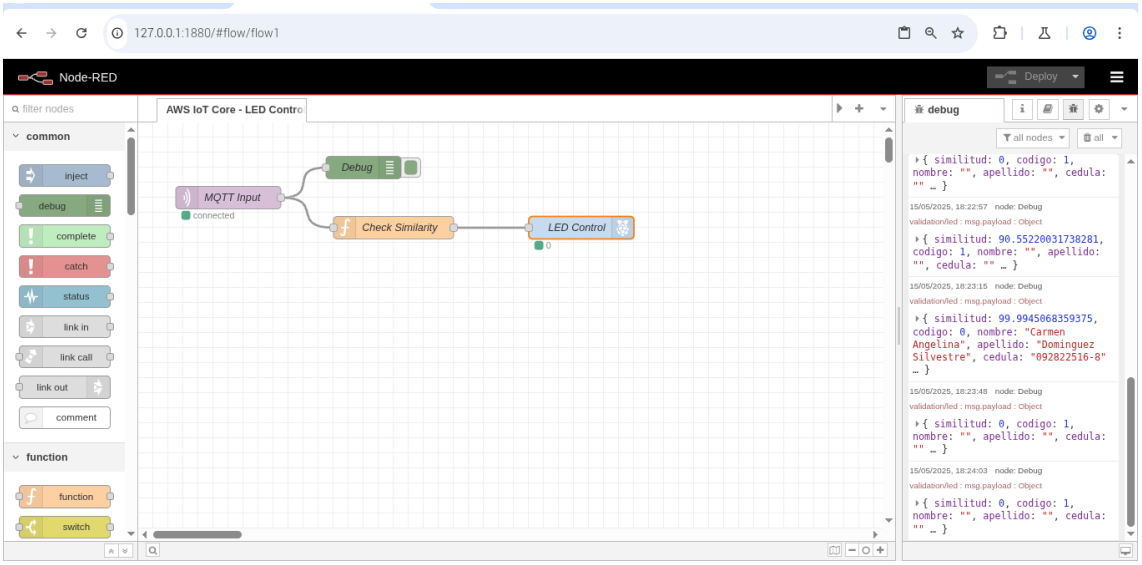

Figura 88. Prueba con una persona que no se encuentran en la Base de Datos

CAPITULO IV - RESULTADOS

4.1. Resultados de la implementación del sistema de seguridad

4.1.1. Diagrama de flujo del análisis de cada fotografía

Se hizo el respectivo análisis de cada fotografía que iría en la base de datos, subiendo por medio de código a Amazon Rekognition en la que proporciona el análisis de cada fotografía.

Usuario 1

En el análisis del usuario 1 con nombre de Carmen Dominguez se obtiene lo siguiente: el faceId, análisis de característica faciales, validación de calidad de imagen y dirección de los ojos. A continuación, se detallará algunos puntos del análisis (ver figura 89)

FaceId: 6b4e8337-6cff-41a6-ad7a-da7fa1ea8ec5

Análisis de característica faciales

- Edad aproximada: 25 - 33 años
- Género: Femenino (99.8%)
- Emoción principal: Calma (92.6%)

Validación de calidad de imagen

- Brillo: 71.70

Dirección de los ojos

- Yaw: 2.4779200553894043
- Pitch: -2.3596293926239014

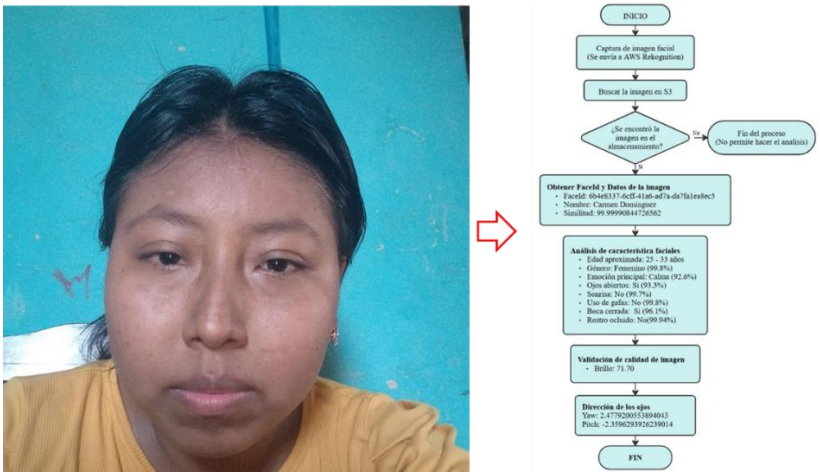

Figura 89. Fotografía subida y análisis de Amazon Rekognition del usuario 1

Usuario 2

En el análisis del usuario 2 con nombre de Ketzia Villón se obtiene lo siguiente: el faceId, análisis de característica faciales, validación de calidad de imagen y dirección de los ojos. A continuación, se detallará algunos puntos del análisis (ver figura 90).

FaceId: 802a161d-cfd9-4965-8d9c-d565680bc239

Análisis de característica faciales

- Edad aproximada: 9-12 años
- Género: Femenino (99.91%)
- Emoción principal: Calma (99.7%)

Validación de calidad de imagen

- Brillo: 71.70

Dirección de los ojos

- Yaw: 3.1143693923950195

- Pitch: -13.400795936584473

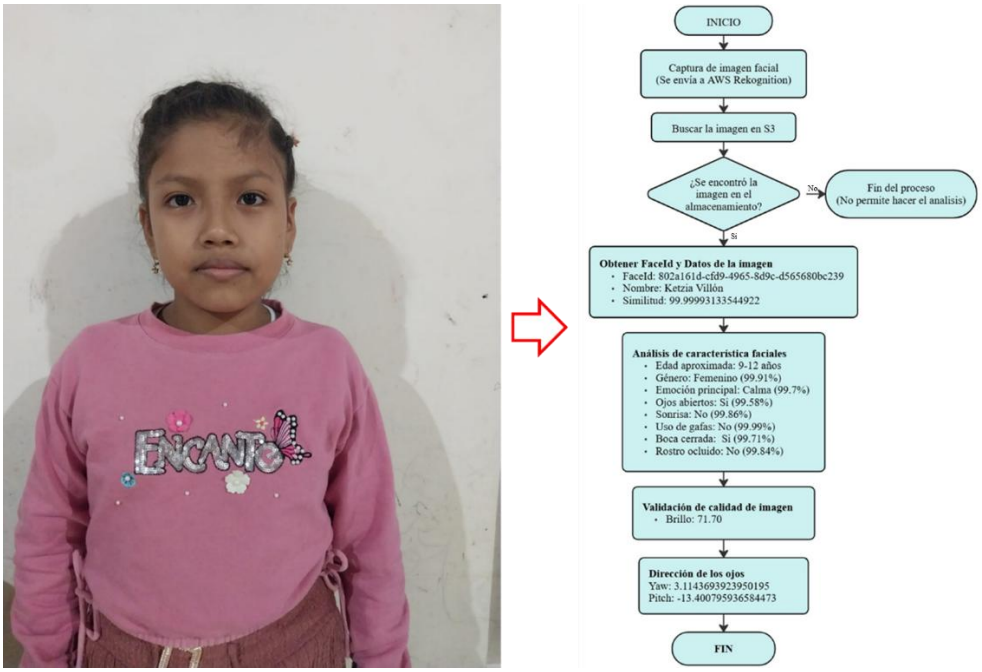

Figura 90. Fotografía subida y análisis de Amazon Rekognition del usuario 2

Usuario 3

En el análisis del usuario 3 con nombre de Daniel Jaramillo se obtiene lo siguiente: el faceId, análisis de característica faciales, validación de calidad de imagen y dirección de los ojos. A continuación, se detallará algunos puntos del análisis (ver figura 91).

FaceId: 5e4da4d0-376a-4d00-ad07-cce42c129c60

Análisis de característica faciales

- Edad aproximada: 31 - 39 años
- Género: Masculino (99.9%)
- Emoción principal: Calma (43.58%)

Validación de calidad de imagen

- Brillo: 71.70

Dirección de los ojos

- Yaw: 0.49805623292922974
- Pitch: 2.380072593688965

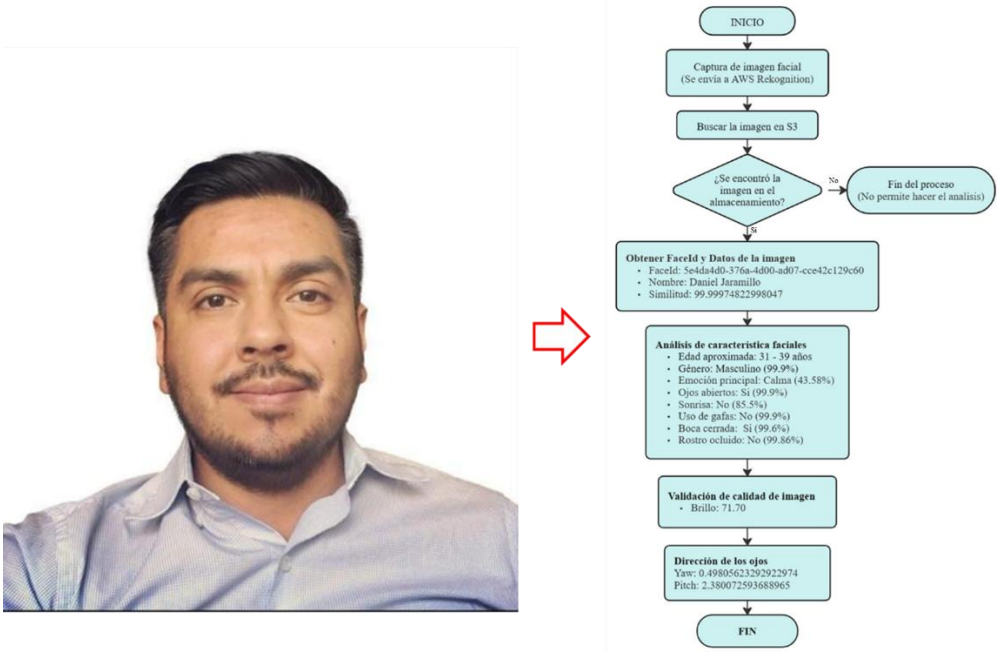

Figura 91. Fotografía subida y análisis de Amazon Rekognition del usuario 3

4.1.2. Registro exitoso de los usuarios en la base de datos

Se logró subir y almacenar los datos personales de cada usuario **(Nombres, Apellidos, Cédula)** y biométricos **(faceId)** en la base de datos DynamoDB y subir cada foto a Amazon Rekognition, se hace mediante AWS CLI. Estas imágenes son almacenas en la colección creada anteriormente llamada **"Colección_10"** generando en el análisis un **faceId** único para enlazar los datos de cada usuario (ver figura 92).

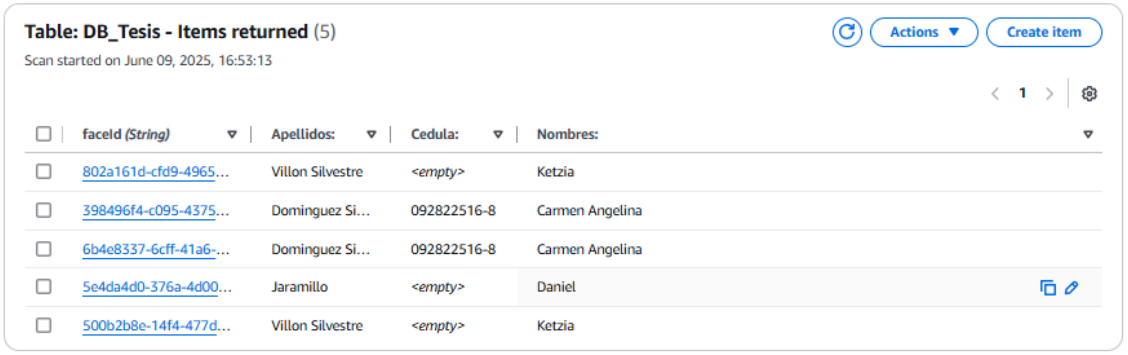

Figura 92. Datos de todos los usuarios en DynamoDB

4.2. Prototipo del sistema de seguridad basado interno de reconocimiento facial con una plataforma de servicio inteligente

El sistema autentica a usuarios autorizados con el resultado de Node-RED que es enviado al Raspberry Pi, el dato proporcionado es recibido y enviado al relé para así activar la cerradura . Ofreciendo una solución segura y eficiente para entornos como residencias inteligente u oficinas, con potencial para mejoras a futuras(ver figura 93).

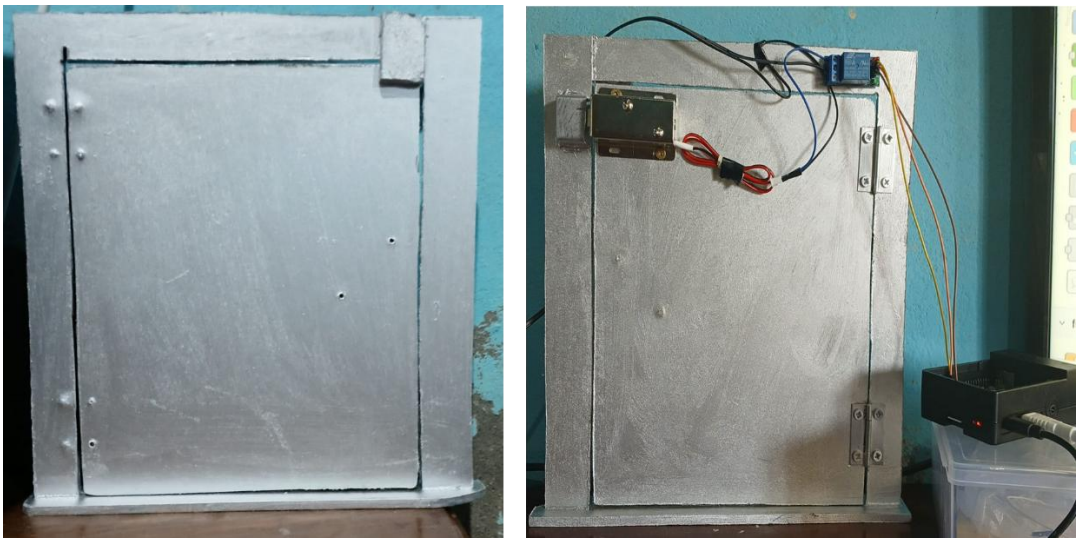

Figura 93. Prototipo de sistema de seguridad interno

4.3. Visualización de trazabilidad con AWS X-Ray

Permite abrir un mapa interactivo que sigue cada paso dependiendo de la petición del cliente a cada servicio. Inicia con el cliente va a una función Lambda llamada **"validacionrostros2"** que es la encargada de procesar la solicitud, interactuara con 3 servicios principales:

- Amazon Rekognition: Análisis de la imagen
- AWS IoT: Crear un broker MQTT para transmitir datos a Node-RED
- DynamoDB: Consultar datos almacenados

Se observa el comportamiento de la arquitectura serveless y permite identificar errores o posibles cuellos de botellas por el monitoreo que proporciona X – Ray (ver figura 94).

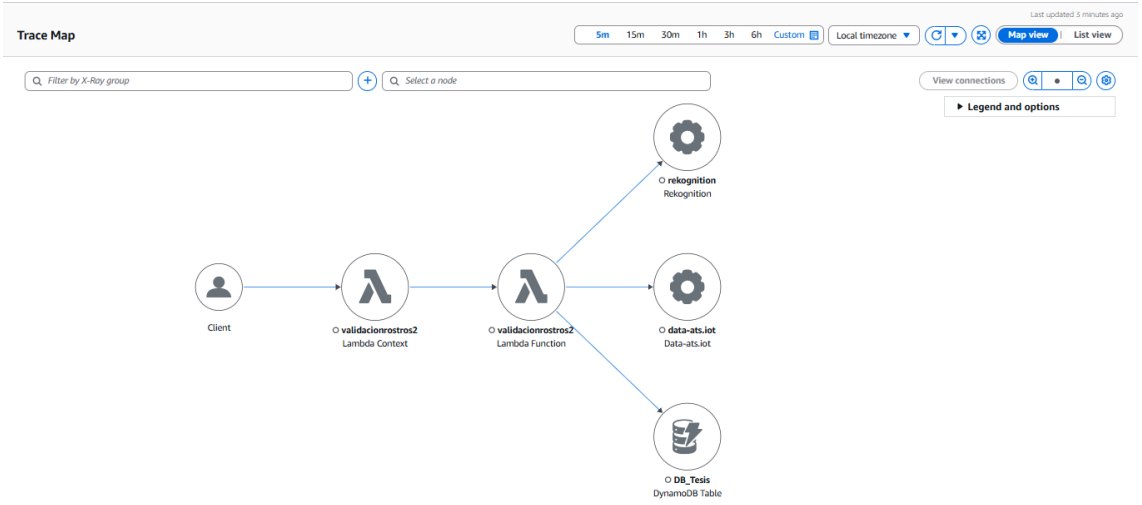

Figura 94. Proceso de trazabilidad con AWS X-Ray

4.4. Resultados del sistema de reconocimiento facial implementado

Para obtener unos resultados se hace la prueba con usuarios registros y no registrados en la base de datos, a través del interfaz HTML conectado con el API Gateway de AWS. los rostros

que fueron capturados son procesados por AWS Rekognition y envía la respuesta por medio de MQTT a Node-RED.

4.4.1. Cuando se encuentra coincidencia y el sistema otorga acceso

Validación de reconocimiento facial en la interfaz

En la interfaz se presenta en el lado izquierdo la cámara en tiempo real y en el lado derecho la imagen capturada. Al momento de realizar la comparación, el sistema da como resultado que la coincidencia es de **100% de similitud**, esto indica que la persona que está siendo verificada se encuentra en la base de datos. Los resultados de validación son:

Nombres: Carmen Angelina

Apellidos: Dominguez Silvestre

Cédula: 092822516-8

Similitud: 100.00%

Descripción: Persona valida detectada

Con los siguientes resultados de validación indica que el sistema logró identificar correctamente a la persona y permitiendo el acceso. La alta precisión del algoritmo es evidente en el grado de similitud alcanzado y en la precisión de los sistemas de respuestas automatizados (ver figura 95).

Figura 95. Usuario registrado - Acceso otorgado

Resultados de la interfaz a Node-RED por MQTT en IoT Core

Luego que se refleje el resultado en la interfaz HTML con similitud de 100%, tal resultado es enviado a IoT Core donde se creó un Client MQTT quien es encargado de enviar dicho resultado a Node-RED por medio un topic **"validation/led"**. Se observa en el terminal de Node-RED los resultados de validación (ver figura 96):

Similitud: 99.99690246582031

Datos: 1

Nombres: Carmen Angelina

Apellidos: Dominguez Silvestre

Cédula: 092822516-8

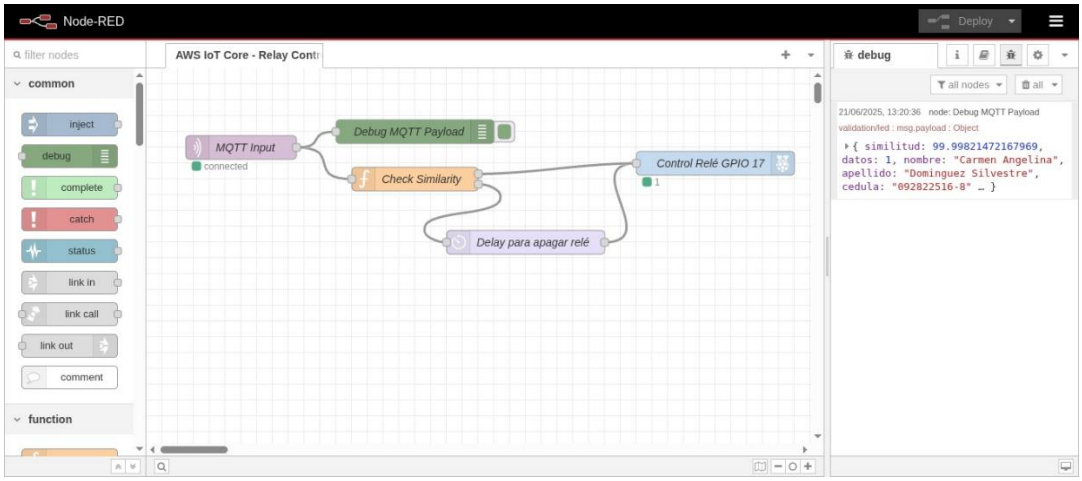

Figura 96. Resultados de Node-RED – Acceso Otorgado

Resultados en el prototipo

Con "1" de Node-RED es recibido por el Raspberry pi dicho 1 que hará activar el led y así dando apertura a la puerta (abre la cerradura)(ver figura 97).

Figura 97. Acceso al usuario – Abrir la cerradura

4.4.2. Cuando no se encuentra coincidencia y el sistema no otorga acceso

Validación de reconocimiento facial en la interfaz

Tras hacer la respectiva comparación, el sistema da como resultado que no existe coincidencia con la imagen captura, esto indica que esa persona no se encuentra en la base de datos. Los resultados de validación son:

Nombres: (vacío)

Apellidos: (vacío)

Cédula: (vacío)

Similitud: 0%

Descripción: No se detectó coincidencia

Al momento en que no es encontrado en la base de datos, niega el acceso. Este escenario es uno de los más importante para garantizar la seguridad y la fiabilidad del sistema, ya que solo las personas registradas en la base de datos tendrán el acceso. Con estos resultados el sistema es capaz de distinguir a las personas registrada y no registrada (ver figura 98).

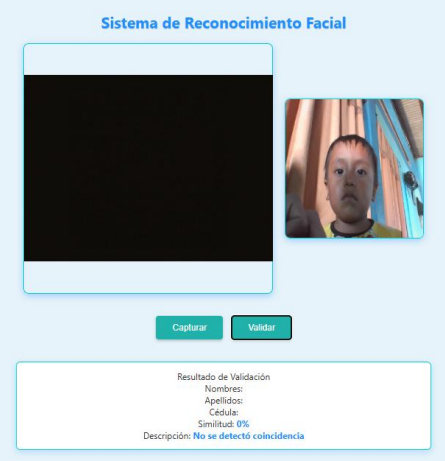

Figura 98. Usuario no registrado - Acceso negado

Resultados de la interfaz a Node-RED por MQTT en IoT Core

El resultado de la interfaz está vacío todos los campos es por lo que no se encontró coincidencia, este resultado es enviado a IoT Core donde se creó un Client MQTT quien va a enviar ese mensaje a Node-RED. Se observa en el terminal de Node-RED los resultados de validación (ver figura 99).

Similitud: 0

Nombres: (vacío)

Apellidos: (vacío)

Datos: 0

Cédula: (vacío)

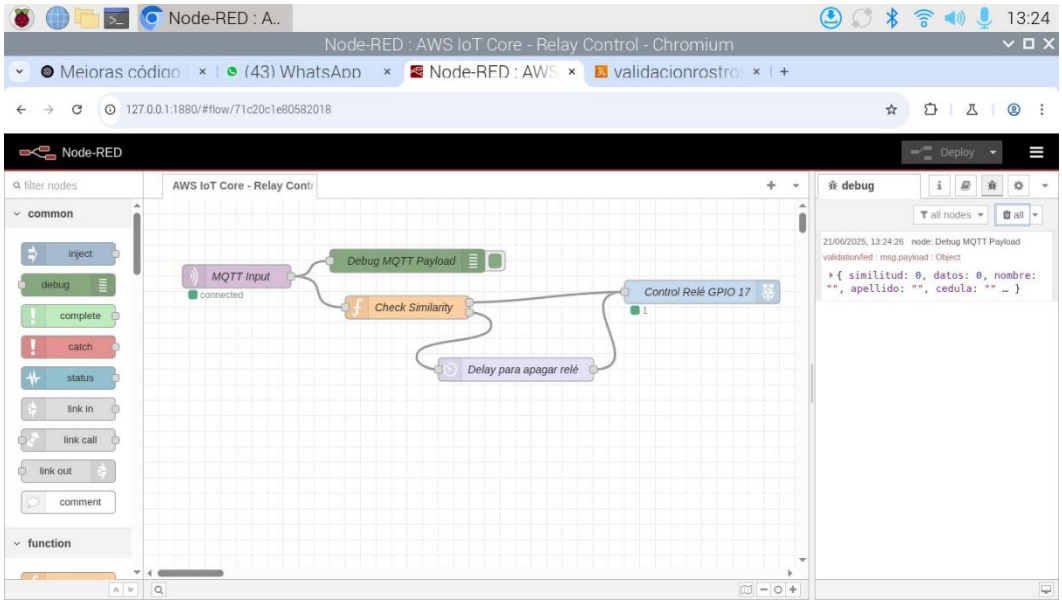

Figura 99. Resultados de Node-Red – Acceso negado

Resultados en el prototipo

Con “0” de Node-RED es recibido por el Raspberry pi dicho 0 que no permitirá activar el relé en la que no dará apertura a la puerta (no se abre la cerradura)(ver figura 100).

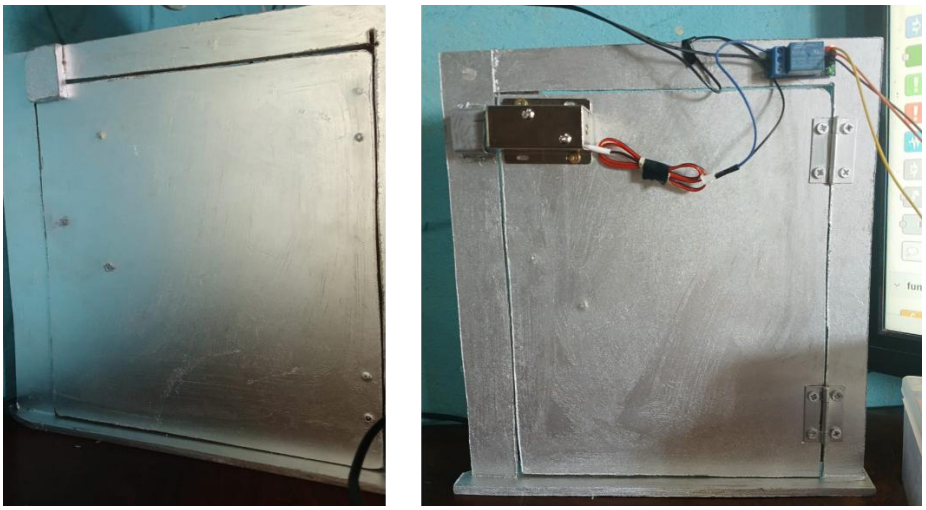

Figura 100. Acceso negado - No abrir la puerta

4.5. Evidencias del funcionamiento en AWS Lambda

Se presentan las gráficas que se obtuvo desde la consola de monitoreo de Lambda en AWS, en la que se invocó la función **“validacionrostros2”** las que evidencian un funcionamiento correctamente de la función que se creó.

Invocaciones son las acciones que se hace al momento en que se envía a validar una fotografía.

4.5.1. Registro de invocaciones

Se muestra la cantidad de invocaciones que se realizó a la función Lambda en un cierto periodo de tiempo, desde las 14:00 hasta aproximadamente 14:35 no se registró invocaciones, por ese motivo sus valores son 0. Pero a partir de las 14:40, se empieza hacer invocaciones en

las que alcanza un pico de 12 invocaciones cerca de las 14:45. Indica que si hubo actividad en un horario especifico (ver figura 101).

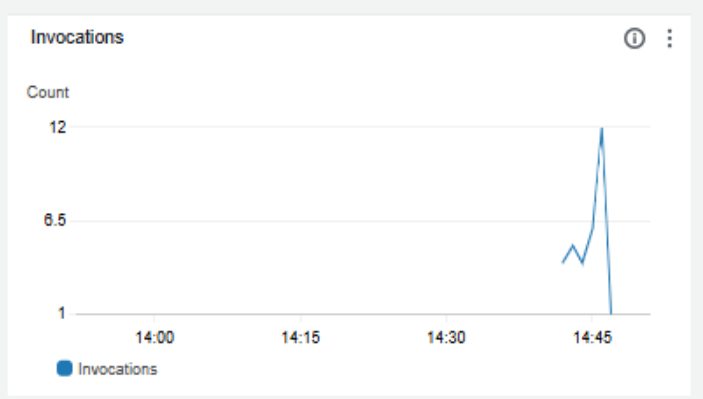

Figura 101. Registro de invocaciones

4.5.2. Tiempo de duración de la ejecución

En la figura 102 se muestran los tiempos de ejecución de todas las invocaciones realizadas entre las 14:00 hasta las 14:45, en las que se observa incremento en un cierto tiempo.

Comportamiento por tiempo específicos

14:00 a 14:40 → En este lapso no se registró ninguna ejecución en la que no muestra ningún dato.

A partir de las 14:40 → Aquí empiezan los registro y datos

Tiempo máximo inicia con un valor superior a 310 ms, su declinación es poco a poco, aunque existen saltos menores se evidencia cambios durante la ejecución.

Tiempo promedio está cerca de los 150 ms, se mantiene casi plana con ligeras oscilaciones alrededor de 189 ms.

Tiempo mínimo inicia alrededor de los 80 ms, se presentan picos y caídas entre los 65 ms y 120 ms apuntando a una variabilidad notable mínimo.

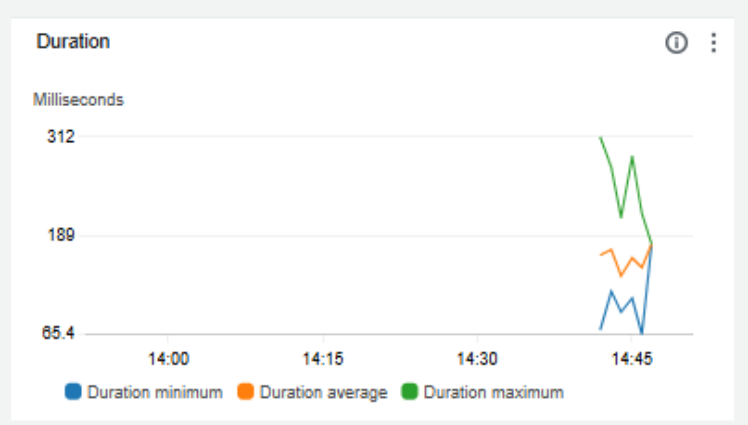

Figura 102. Tiempo de duración de la ejecución

4.5.3. Tasa de éxito y registro de errores

En la figura 103 se muestran los porcentajes de éxitos y los errores, desde las 14:40 hasta las 14:45, la tasa de éxito permanece en 100%, sin reflejar errores en las que el contador está en 0%. Como resultado se obtiene que todas las invocaciones funcionaron correctamente junto al sistema durante el periodo de actividad.

Figura 103. Tasa de éxito y registro de errores

4.6. Registro detallado de invocaciones y consumo de recursos en Lambda

4.6.1. Registro detallado de invocaciones

Las invocaciones que se hizo en la función Lambda se pueden observar a más detalle de cada uno en donde se ve el tiempo de ejecución, el tiempo facturado, la memoria asignada y la

memoria utilizada (ver figura 104). A continuación, los resultados dado por Lambda se detallarán:

#(Numero de orden): Se observa 9 invocaciones recientes, numeradas del 1 al 9.

Timestamp (Marca de tiempo): La fecha en la que se hizo la invocación fue el 16 de junio del 2025, desde las 14:53 hasta las 15:01 horas.

- Invocación 1: 15:01:30 (último)
- Invocación 2: 14:57:43
- Invocación 3: 14:57:32

RequestId (ID de solicitud): Cada ejecución que se haga tendrá un identificador único que generará AWS. Es muy importante este ID porque va a permitir consultar todos los registros específicos de cada ejecución.

- Invocación 1: 38ad011e-4008-4984-b466-a37ef5395c8b
- Invocación 2: 9bbd552a-f5c8-4838-a9ad-e394cbb9287e
- Invocación 3: cc00c005-6ee0-4a54-ad9d-ec06b56eb7e9

LogStream (flujo de registro): Cada invocación que se haga en diferentes momentos tendrá asociado un enlace de flujo de registro, la que permitirá acceder a CloudWatch. Para este caso, todas las invocaciones que se hizo Lambda proporciono el siguiente enlace de flujo **2025/06/16/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9**.

Duración en milisegundos (DurationInMS): En esta columna se observará el tiempo de ejecución que se llevó cada invocación, entre las nueve (9) invocaciones se destaca el tiempo

máximo y el mínimo. La variación de los tiempos se debe a la diferencia de tamaño de cada imagen procesada.

- Duración Máxima: 330.48 ms (primera invocación)
- Duración Mínima: 100.36 ms (cuarta invocación)

Duración facturada en Milisegundos (BilledDurationInMS): AWS factura todas las invocaciones por múltiplos de milisegundos, para esto se redonde hacia arriba. En la que se verá cual fue la invocación más costosa y la menos costosa.

- Más costosa: 331.0 ms (primera invocación)
- Menos costosa: 101.0 ms (cuarta invocación)

Memoria asignada (MemorySetInMB): La memoria asignada para esta función de Lambda fue de **512 MB**, se configuró para así garantizar un buen desempeño de la función, en especial al momento de que la imagen se procese en tiempo real.

Memoria utilizada (MemoryUsedInMB): El tamaño real de la memoria fue de 96 MB para todas las invocaciones que se hizo en ese momento. Esto demuestra que los 512 MB configurados son más que suficiente al momento de procesar una imagen ya que su consumo real es de aproximadamente el **18.75%** de la memoria configurada.

Recent invocations

#	Timestamp	RequestId	LogStream	DurationInMS	BilledDurationInMS	MemorySetI...	MemoryUsedIn...
1	2025-06-09T14:51:06.131-05:00	38ad011e-4008-4984-b466-a37ef5395c8b	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	330.48	331.0	512.0	96.0
2	2025-06-09T14:47:53.037-05:00	9bbd552a-f5c5-4838-a9ad-e394cbb9287e	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	178.47	179.0	512.0	95.0
3	2025-06-09T14:46:31.698-05:00	cc00c005-6ee0-4a54-ad9d-ec06b56eb7e9	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	216.13	217.0	512.0	95.0
4	2025-06-09T14:46:29.166-05:00	c515cb76-ef7a-48af-b6ff-8d02715e12c3	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	100.36	101.0	512.0	95.0
5	2025-06-09T14:46:27.547-05:00	c46b2b69-3924-4850-9f7a-7bf88c38067e	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	161.91	162.0	512.0	95.0
6	2025-06-09T14:46:25.973-05:00	c9d877d1-a98c-4a08-b578-c01fc246ffed	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	143.76	144.0	512.0	95.0
7	2025-06-09T14:46:23.515-05:00	c7884223-a43f-4bab-b6f8-ecad2f3d35f8	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	190.9	191.0	512.0	95.0
8	2025-06-09T14:46:21.616-05:00	bee2de35-8001-433e-9c5a-1c07471eea4f	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	137.19	138.0	512.0	95.0
9	2025-06-09T14:46:19.815-05:00	f95ede83-9172-4901-afb3-7b640024ca03	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	148.31	149.0	512.0	95.0

Figura 104. Registro de invocaciones

4.6.2. Invocaciones más costosas en GB-segundos

En esta sesión se detallará las invocaciones que representan con mayor costo, considerante la relación entre la memoria asignada y la memoria facturada (la memoria utilizada), el análisis que se observa es para identificar que invocaciones generan mayor consuma de recursos (ver figura 105). A continuación , se detallará el análisis.

#(Numero de orden): Se observa 9 invocaciones recientes, que estarán ordenadas de mayor a menor consumo .

Timestamp (Marca de tiempo): Las invocaciones se hizo entre las 14:53 y las 15:01 del día 16 de junio del 2025.

- Invocación 1: 14:51:06 (último)
- Invocación 2: 14:42:16
- Invocación 3: 14:45:40

RequestId (ID de solicitud): Cada invocación que se haga tendrá un identificador único que generará AWS, en la que es esencial para realizar trazabilidad y consultar los registros específicos.

- Invocación 1: 38ad011e-4008-4934-bd60-a37ef3963c0b
- Invocación 2: 94ce3e0e-0684-4538-988e-fcf84b7fecb8
- Invocación 3: 6e9c0162-74f5-47a2-bee9-4705374781c0

LogStream (flujo de registro): Para esta ocasión tiene el mismo enlace anterior proporcionado por Lambda que es el siguiente **2025/06/16/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9**.

Duración facturada en Milisegundos (BilledDurationInMS): AWS factura todas las invocaciones por múltiplos de milisegundos en las que estos resultados se van a redondear para arriba, estos tiempos oscilan entre:

- Más costosa: 331.0 ms (primera invocación)
- Menos costosa: 179.0 ms (última invocación)

Duración facturada en GB-segundos (BilledDurationInGBSeconds): Para obtener este valor se obtiene multiplicando la memoria asignada (GB) por el tiempo facturado (milisegundo).

Datos

$$Memoria\ asignada\ =\ 512\ MB\ *\ 0.001\ =0.5\ GB$$

$$Tiempo\ facturado\ máximo\ =\ 331\ ms$$

$$Tiempo\ facturado\ mínimo\ =\ 179\ ms$$

Mayor consumo

$$\boldsymbol{Duración\ facturado = memoria\ asignada(GB) * tiempo\ facturado\ (ms)}$$

$$Duracion\ facturado = 0.5\ GB * 331\ ms$$

$$Duración\ facturado = 165.5\ GBms$$

$$Duración\ facturado = 0.1655\ GB - segundos$$

Menor consumo

$$Duración\ facturado = memoria\ asignada(GB) * tiempo\ facturado\ (ms)$$

$$Duracion\ facturado = 0.5\ GB * 179\ ms$$

$$Duración\ facturado = 89.5\ GBms$$

$$Duración\ facturado = 0.0895\ GB - segundos$$

Most expensive invocations in GB-seconds (memory assigned * billed duration)

#	Timestamp	RequestId	LogStream	BilledDurationInMS	MemorySetI...	BilledDurationInGBSeconds
1	2025-06-09T14:51:06.131-05:00	38ad011e-4008-4984-b466-a37ef5395c8b	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	331.0	512	0.1655
2	2025-06-09T14:42:16.359-05:00	94ce3e0e-0682-4538-988e-fcf84b7fecb8	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	313.0	512	0.1565
3	2025-06-09T14:45:40.114-05:00	6e9c0162-74f5-47a2-bee9-470537478 1c0	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	288.0	512	0.144
4	2025-06-09T14:43:57.077-05:00	3e9c72b0-f69f-4fa3-bfe0-7b310ab20705	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	274.0	512	0.137
5	2025-06-09T14:46:31.698-05:00	cc00c005-6ee0-4a54-ad9d-ec06b56eb7e9	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	217.0	512	0.1085
6	2025-06-09T14:44:20.278-05:00	d38945b1-359d-4f3a-bca1-961c0a308024	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	212.0	512	0.106
7	2025-06-09T14:46:23.515-05:00	c7884223-a43f-4bab-b6f6-ecad2f3d35f8	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	191.0	512	0.0955
8	2025-06-09T14:43:02.674-05:00	135e855e-bba0-4705-8996-ada01628f999	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	190.0	512	0.095
9	2025-06-09T14:47:53.037-05:00	9bbd552a-f5c5-4838-a9ad-e394cbb9287e	2025/06/09/[$LATEST]5c0f66677de744a8a99d1ec9c0c98bb9	179.0	512	0.0895

Figura 105. Registro de invocaciones y consumo de registro

CONCLUSIONES

- De acuerdo con el estado de arte se puede evidenciar que la mayor parte de la investigación que se realizó es a través de aprendizaje profundo y de los cuales uno de los algoritmos tratado es FaceNet, AWS Rekognition, con registros altos para la seguridad interna.
- La selección de los servicios tecnológicos utilizados cumple con lo requerido para el sistema de reconocimiento facial. Siendo eficiente, seguro y escalable. La integración de diversos servicios de Amazon Web Services, AWS Rekognition se convirtió en principal para el análisis facial, DynamoDB facilitó la administración de datos, IoT Core ayudó a controlar dispositivos físicos desde la nube, Node -RED sirvió como plataforma visual para manejar los flujos de mensajería MQTT, se creó un sistema flexible.
- Las pruebas del sistema demostraron una alta precisión y efectividad en la identificación de usuarios registrados, con coincidencias en la base de datos con una similitud del 100%. En escenarios con usuarios no registrados, el sistema negó el acceso con una similitud del 0%, lo que valida su capacidad para brindar seguridad. La combinación de AWS IoT Core, Node-RED y Raspberry Pi proporcionó una respuesta en tiempo real al abrir la cerradura eléctrica solo para usuarios autorizados
- Las métricas recolectadas desde AWS Lambda, como una tasa de éxito del 100% y un tiempo promedio de ejecución de 150 milisegundos, ratificaron el alto desempeño y confiabilidad del sistema en entornos reales. Estos resultados

consolidan la viabilidad de implementar soluciones inteligentes de seguridad interna basadas en reconocimiento facial y servicios en la nube.

RECOMENDACIONES

- Explorar otras plataformas que permitan entrenar modelos de reconocimiento facial, como Google Cloud Vision o Microsoft Azure, con la finalidad de comparar el tiempo de respuesta, costos y su precisión con AWS Rekognition.
- Explorar más servicios que proporciona AWS, como AWS SNS permitirá enviar notificaciones de manera automática por SMS o correo cuando detecte a una persona que no se encuentra en la base de datos y AWS Cognito este servicio permite controlar el acceso de los usuarios a la interfaz.
- Optimizar el sistema para lugares con poca iluminación, esto se logra con un procesamiento adicional a la imagen como normalizar el brillo y contraste. Hacer los respectivos ajustes en Amazon Rekognition mejoraría el sistema.
- Evaluar con un módulo adicional de comunicación, como es LoRa o NB-IoT para así extender un sistema de cobertura limitada o para un entorno difícil de tener conexión a una red Wi-Fi.
- Actualizar cada cierto tiempo la base de datos (DynamoDB), ver todas las actualizaciones de AWS a cada servicio y reforzar las políticas de seguridad en AWS para evitar que el sistema deje de funcionar.
- Realizar pruebas a mayor escala probando en otros entornos como lo es una aula, laboratorios o empresas. En la que si será necesario aumentar la capacidad de la colección en AWS Rekognition el aumento de capacidad tendría un costo adicional.

BIBLIOGRAFIAS

[1] Kevin Ortega and Sergio Pino, “Impacto social y económico de los factores de riesgo que afectan la seguridad ciudadana en Ecuador,” Revistas Espacios , pp. 1–19, 2021, [Online]. Available: https://revistaespacios.com/a21v42n21/21422104.html

[2] Cesar Armando Nacipucha Nacipucha and Jonathan Joel Frías Pérez, “Diseño de un prototipo de control de acceso a traves de reconocimiento facial mediante la creación de un algoritmo basado en software libre utilizando LattePanda,” Universidad Politecnica Salesiana del Ecuador, Guayaquil, 2020. [Online]. Available: https://dspace.ups.edu.ec/bitstream/123456789/19626/1/UPS-GT003086.pdf

[3] Bárbara María Cabrera Castro, “LOCALIZACIÓN PRECISA DE ACTIVOS Y HUMANOS EN ESPACIOS CERRADOS PARA SEGURIDAD EN ENTORNOS INDUSTRIALES,” UNIVERSITAT POLITÈCNICA DE VALÈNCIA, Valencia, 2020. [Online]. Available: https://riunet.upv.es/handle/10251/159399

[4] MARÍA DEL MAR PLAZA CANO, “ESTUDIO Y DESARROLLO DE UN SISTEMA DE LOCALIZACIÓN PRECISA DE ACTIVOS Y HUMANOS EN ESPACIOS CERRADOS PARA SEGURIDAD EN ENTORNOS CON INTERACCIÓN HUMANO ROBOT,” UNIVERSITAT POLITÈCNICA DE VALÈNCIA, Valencia, 2021. [Online]. Available: https://riunet.upv.es/handle/10251/174614

[5] FRANKLIN ARMANDO VACA PIÑA and FAUSTO GERMAN RIVERA RODRÍGUEZ, “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE ACCESO MEDIANTE RECONOCIMIENTO FACIAL PARA LA ACADEMIA

TITANES CUENCA," UNIVERSIDAD POLITÉCNICA SALESIANA, Cuenca , 2022. [Online]. Available: http://dspace.ups.edu.ec/handle/123456789/24611

[6] Diego Fernando Legarda Delgado and Oscar Andrés Loaiza Pabón, "RECONOCIMIENTO FACIAL PARA LA AUTOMATIZACIÓN DEL REGISTRO DE ASISTENCIA A CLASES.," Universidad Tecnológica de Pereira, Pereira , 2022. [Online]. Available: https://hdl.handle.net/11059/14172

[7] J. J. Domínguez Espinoza and L. F. Domínguez Espinoza, "Implementación de un sistema de video vigilancia con reconocimiento facial como servicio alternativo a la seguridad física en una empresa de la ciudad de Guayaquil.," ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL, Guayaquil , 2023. [Online]. Available: http://www.dspace.espol.edu.ec/handle/123456789/62798

[8] PROTEK, "¿Qué importancia tiene la seguridad en las instituciones educativas? ," PROTEK. [Online]. Available: https://www.protek.com.py/novedades/que-importancia-tiene-la-seguridad-en-las-instituciones-educativas

[9] Mario Rodriguez, "SEGURIDAD INSTITUCIONAL," Slideshare. [Online]. Available: https://es.slideshare.net/slideshow/trabajo-hoy-lunes/46992750

[10] TIC TAC TECH, "SEGURIDAD EN ESCUELAS: 6 MEDIDAS QUE NO PUEDEN FALTAR," TIC TAC TECH. [Online]. Available: https://pyv.technology/blog/seguridad-en-escuelas

[11] lonix, "Qué es la biometría: tipos, ventajas y ejemplos de uso," lonix. [Online]. Available: https://ionixlatam.com/que-es-la-biometria-tipos-ventajas-y-ejemplos-de-uso

[12] Cristina Domingo Jaramillo, “Utilización del sistema de reconocimiento facial para preservar la seguridad ciudadana,” El criminalista digital - Papeles de criminologia, vol. 9, no. 2340–6046, pp. 20–37, Feb. 2021, [Online]. Available: https://revistaseug.ugr.es/index.php/cridi/article/view/20899

[13] Biometrics, “Identificación biométrica a través del iris ocular,” Biometrics. [Online]. Available: https://biometrics-on.com/identificacion-biometrica-a-traves-del-iris-ocular

[14] Samuel Kellett, “Tecnología de reconocimiento facial: lo que debe saber,” Academy. [Online]. Available: https://www.avast.com/es-es/c-facial-recognition

[15] Pablo Huet, “Qué son las redes neuronales y sus aplicaciones,” OpenWebinars. [Online]. Available: https://openwebinars.net/blog/que-son-las-redes-neuronales-y-sus-aplicaciones

[16] AWS, “Reconocimiento Facial ,” Amazon. [Online]. Available: https://aws.amazon.com/what-is/facial-recognition

[17] Juan Francisco Diaz y Diaz, “Reconocimiento facial: FaceNet y AWS Rekognition,” Pontificia Universidad Catolica Argentina, Argentina, 2020. [Online]. Available: https://repositorio.uca.edu.ar/bitstream/123456789/11212/1/reconocimiento-facial-facenet-aws.pdf

[18] Raúl Ignacio Vásquez Núñez, “GENERACION DE APLICACIÓN QUE PERMITA EL RECONOCIMIENTO DE UNA CELEBRIDAD MEDIANTE LA INTEROPERABILIDAD DE SERVICIOS DE AMAZON WEB SERVICES PARA LA

EMPRESA ‘CONTINUUM ET COMPAGNIE’ ,” UNIVERSIDAD ANDRÉS BELLO , SANTIAGO, 2021. [Online]. Available: https://repositorio.unab.cl/xmlui/handle/ria/21973

[19] AWS, “Amazon S3,” Amazon. [Online]. Available: https://aws.amazon.com/es/s3

[20] AWS, “Amazon Simple Queue Service,” Amazon. [Online]. Available: https://aws.amazon.com/es/sqs/

[21] AWS, “Amazon Simple Notification Service,” Amazon. [Online]. Available: https://aws.amazon.com/es/sns

[22] Amazon Web Services, “¿Qué es Amazon API Gateway?,” https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/welcome.html.

[23] Amazon Web Services, “¿Qué es AWS Lambda?,” https://docs.aws.amazon.com/es_es/lambda/latest/dg/welcome.html.

[24] Amazon Web Services, “¿Qué es Amazon DynamoDB?,” https://docs.aws.amazon.com/es_es/amazondynamodb/latest/developerguide/Introduction.html.

[25] Amazon Web Services, “Connect to AWS IoT Core,” https://docs.aws.amazon.com/es_es/iot/latest/developerguide/connect-to-iot.html.

[26] Datademia, “¿Qué es Google Cloud Platform?,” Datademia. [Online]. Available: https://datademia.es/blog/que-es-google-cloud-platform#:~:text=Google%20Cloud%20Platform%20es%20un,big%20data%2C%20entre%20otras%20funcionalidades

[27] Ben-Lutkevich, “Google Compute Engine,” Cloud Computing. [Online]. Available: https://www.techtarget.com/searchcloudcomputing/definition/Google-Compute-Engine

[28] Google Cloud, “¿Qué es Cloud Storage?,” Google Cloud. [Online]. Available: https://cloud.google.com/learn/what-is-cloud-storage?hl=es-419

[29] Google cloud platform, “Google Kubernetes Engine,” Geeksforgeeks. [Online]. Available: https://cloud.google.com/kubernetes-engine?hl=es_419

[30] Servinfomracion, “Cloud SQL en GCP,” Servinfomracion. [Online]. Available: https://mesadeayuda.servinformacion.com/knowledge/cloud-sql-en-gcp

[31] IBM, “¿Qué es AutoML?,” IBM. [Online]. Available: https://www.ibm.com/es-es/think/topics/automl#:~:text=El%20machine%20learning%20automatizado%20(AutoML,machine%20learning%20(modelos%20ML).

[32] Google Cloud, “Dialogflow,” Google Cloud . Accessed: Apr. 29, 2025. [Online]. Available: https://cloud.google.com/dialogflow/docs?hl=es-419

[33] Cloud Computing, “Google BigQuery: qué es y para qué sirve,” Google Cloud. [Online]. Available: https://cloud.google.com/bigquery/docs/introduction?hl=es-419

[34] Google Cloud, “Instructivo de detección de rostro,” Google Cloud . [Online]. Available: https://cloud.google.com/vision/docs/face-tutorial?hl=es-419

[35] AWS, “¿Qué es MQTT?,” Amazon. [Online]. Available: https://aws.amazon.com/es/what-is/mqtt

[36] “Processors - Raspberry Pi Documentation,” https://www.raspberrypi.com/documentation/computers/processors.html#bcm2711.

ANEXOS

Anexo 1 Código de la política para el buckets de almacenamiento

```
{
        "Version": "2012-10-17",
        "Statement": [
                {
                        "Sid": "PermitirLambdaPutObject",
                        "Effect": "Allow",
                        "Principal": {
                                "AWS": "arn:aws:sts::820242919138:assumed-
role/validacionrostros2-role-4ucpu1yf/validacionrostros2"
                        },
                        "Action": "s3:PutObject",
                        "Resource": "arn:aws:s3:::fotos-capturas/capturas/*"
                }
        ]
}
```

Anexo 2 Código de la política para DynamoDB

```
{
        "Version": "2012-10-17",
        "Statement": [
        {
                "Effect": "Allow",
                "Action": "dynamodb:GetItem",
                "Resource": "arn:aws:dynamodb:us-east-
2:820242919138:table/DB_Tesis"
                }
        ]
}
```

Anexo 3 Código de la política en IoT Core

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:Receive",
    "iot:PublishRetain"
   ],
   "Resource": "arn:aws:iot:us-east-2:820242919138:topic/*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": "arn:aws:iot:us-east-2:820242919138:topicfilter/*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-2:820242919138:client/ValidationDevice"
  }
 ]
}
```

Anexo 4 Código de Lambda

```
import boto3
import base64
import json
import uuid
from datetime import datetime
# Para activar la instrumentación
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all
patch_all()  # Instrumenta boto3, requests, etc.
rekognition = boto3.client('rekognition', region_name='us-east-2')
iot_client = boto3.client('iot-data', region_name='us-east-2')
dynamodb = boto3.resource('dynamodb', region_name='us-east-2')
s3 = boto3.client('s3', region_name='us-east-2')  # Cliente S3

COLLECTION_ID = 'Coleccion_10'
TABLA_PERSONAS = 'DB_Tesis'
BUCKET_NAME = 'fotos-capturas'
def lambda_handler(event, context):
    try:
        if 'body' in event:
            body = json.loads(event['body'])
        else:
            body = event
        base64_image = body.get('imgvalidacion', '')
        if not base64_image:
            return error_response("No se proporcionó imagen")

        if "base64," in base64_image:
            base64_image = base64_image.split("base64,")[1]

        image_bytes = base64.b64decode(base64_image)
        # Guardar la imagen en S3
        nombre_imagen =
f"capturas/{datetime.now().strftime('%Y%m%d_%H%M%S')}_{str(uuid.uuid4())}.jpg"
        with xray_recorder.in_subsegment('Carga en S3'):
            s3.put_object(
                Bucket=BUCKET_NAME,
                Key=nombre_imagen,
                Body=image_bytes,
                ContentType='image/jpeg'
            )
```

```
# Generar URL firmada para visualizar la imagen
url_foto = s3.generate_presigned_url('get_object',
    Params={'Bucket': BUCKET_NAME, 'Key': nombre_imagen},
    ExpiresIn=3600  # 1 hora de validez
)

with xray_recorder.in_subsegment('Busqueda en Rekognition'):
    response = rekognition.search_faces_by_image(
        CollectionId=COLLECTION_ID,
        Image={'Bytes': image_bytes},
        MaxFaces=1,
        FaceMatchThreshold=70
    )

similitud = 0
datos = 0
nombres = ""
apellidos = ""
cedula = ""
descripcion = "No se detectó coincidencia"

if response['FaceMatches']:
    face_match = response['FaceMatches'][0]
    similitud = face_match['Similarity']
    face_id = face_match['Face']['FaceId']

    with xray_recorder.in_subsegment('Consulta en DynamoDB'):
        tabla = dynamodb.Table(TABLA_PERSONAS)
        resultado_db = tabla.get_item(Key={'faceId': face_id})

    if 'Item' in resultado_db:
        persona = resultado_db['Item']
        nombres = persona.get('Nombres: ', '')
        apellidos = persona.get('Apellidos: ', '')
        cedula = persona.get('Cedula: ', '')
        datos = 1
        descripcion = "Coincidencia detectada"
    else:
        descripcion = "Coincidencia detectada, pero sin datos asociados"
resultado = {
    'similitud': similitud,
    'datos': datos,
    'nombre': nombres,
    'apellido': apellidos,
```

```
            'cedula': cedula,
            'descripcion': descripcion,
            'foto_guardada': nombre_imagen,
            'url_foto': url_foto
        }
        with xray_recorder.in_subsegment('Publicación MQTT en IoT Core'):
            iot_client.publish(
                topic='validation/led',
                qos=1,
                payload=json.dumps(resultado)
            )

        return {
            'statusCode': 200,
            'body': json.dumps(resultado),
            'headers': {
                'Content-Type': 'application/json',
                'Access-Control-Allow-Origin': '*'
            }
        }

    except Exception as e:
        print("Error:", str(e))
        return error_response(str(e))

def error_response(message):
    return {
        'statusCode': 500,
        'body': json.dumps({'error': message}),
        'headers': {
            'Content-Type': 'application/json',
            'Access-Control-Allow-Origin': '*'
        }
    }
```

Anexo 5 Código HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Sistema de Reconocimiento Facial</title>
   <style>
      body {
         font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
         background-color: #e6f3fa;
         display: flex;
         flex-direction: column;
         align-items: center;
         justify-content: center;
         min-height: 100vh;
         margin: 0;
         padding: 20px;
         box-sizing: border-box;
      }
      h2 {
         color: #1e90ff;
         font-size: 28px;
         margin-bottom: 20px;
         text-align: center;
         text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.1);
      }
      .media-container {
         display: flex;
         gap: 20px;
         align-items: center;
         margin-bottom: 20px;
      }
      }
      #video, #canvas {
         width: 450px;
         height: 450px;
         border-radius: 10px;
         box-shadow: 0 4px 12px rgba(0, 123, 255, 0.3);
         border: 2px solid #00ced1;
      }
```

```
        #photo {
    width: 250px;
    height: 250px;
    border-radius: 10px;
    box-shadow: 0 4px 12px rgba(0, 123, 255, 0.3);
    border: 2px solid #00ced1;
    display: none;
}
     .button-container {
    display: flex;
    gap: 15px;
    margin: 20px 0;
}
button {
    padding: 12px 30px;
    font-size: 16px;
    font-weight: 500;
    color: white;
    background-color: #20b2aa;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease, transform 0.2s ease;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
}
button:hover {
    background-color: #008b8b;
    transform: scale(1.05);
}
#result {
    margin-top: 20px;
    padding: 15px;
    background-color: #ffffff;
    border-radius: 8px;
    box-shadow: 0 4px 12px rgba(0, 123, 255, 0.2);
    width: 720px;
    text-align: center;
    color: #333333;
    border: 1px solid #00ced1;
}
#result span {
    font-weight: bold;
    color: #1e90ff;
}
```

```
        #error {
            color: #ff4500;
            margin-top: 10px;
            display: none;
            font-weight: 500;
            padding: 10px;
            background-color: #ffebee;
            border-radius: 5px;
            border: 1px solid #ff4500;
        }
    </style>
</head>
<body>
    <h2>Sistema de Reconocimiento Facial</h2>

    <div class="media-container">
        <video id="video" autoplay></video>
        <canvas id="canvas" style="display:none;"></canvas>
        <img id="photo" src="" alt="Foto capturada">
    </div>

    <div class="button-container">
        <button onclick="capturePhoto()">Capturar</button>
        <button onclick="validatePhoto()">Validar</button>
    </div>

    <div id="result">
        Resultado de Validación<br>
        Nombres: <span id="nombre"></span><br>
        Apellidos: <span id="apellido"></span><br>
        Cédula: <span id="cedula"></span><br>
        Similitud: <span id="similarity"></span><br>
        Descripción: <span id="description"></span>
    </div>
    <div id="error"></div>

    <script>
        let stream;
        const video = document.getElementById('video');
        const canvas = document.getElementById('canvas');
        const photo = document.getElementById('photo');
```

```
const result = document.getElementById('result');
const nombre = document.getElementById('nombre');
const apellido = document.getElementById('apellido');
const cedula = document.getElementById('cedula');
const similarity = document.getElementById('similarity');
const description = document.getElementById('description');
const errorDiv = document.getElementById('error');
const apiUrl = 'https://k5g7c8s5pi.execute-api.us-east-2.amazonaws.com/test';

async function initCamera() {
    try {
        stream = await navigator.mediaDevices.getUserMedia({ video: true });
        video.srcObject = stream;
        video.play();
    } catch (err) {
        console.error("Error al acceder a la cámara: ", err);
        showError("Error al acceder a la cámara: " + err.message);
    }
}

function capturePhoto() {
    const context = canvas.getContext('2d');
    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;
    context.drawImage(video, 0, 0, canvas.width, canvas.height);
    const imageDataUrl = canvas.toDataURL('image/jpeg');
    photo.src = imageDataUrl;
    photo.style.display = 'block';
    errorDiv.style.display = 'none';
}
function showError(message) {
    errorDiv.textContent = message;
    errorDiv.style.display = 'block';
}

async function validatePhoto() {
    if (!photo.src || photo.src === window.location.href) {
        showError('No se ha capturado ninguna foto');
        return;
    }
    try {
        const capturedBase64 = await imageToBase64(photo);

        const response = await fetch(apiUrl, {
            method: 'POST',
            headers: {
```

```
                'Content-Type': 'application/json'
              },
              body: JSON.stringify({
                imgvalidacion: capturedBase64
              })
            });

            if (!response.ok) {
              const errorText = await response.text();
              throw new Error(`Error en la solicitud a la API: ${response.status} -
${errorText}`);
            }

            const data = await response.json();
            console.log("Respuesta de la API:", data);

            errorDiv.style.display = 'none';

            if (data.similitud > 0 && data.datos === 1) {
              nombre.textContent = data.nombre || '';
              apellido.textContent = data.apellido || '';
              cedula.textContent = data.cedula || '';
              similarity.textContent = `${data.similitud.toFixed(2)}%`;
              description.textContent = data.descripcion || 'Persona válida detectada';
            } else {
              nombre.textContent = '';
              apellido.textContent = '';
              cedula.textContent = '';
              similarity.textContent = '0%';
              description.textContent = data.descripcion || 'No se pudo validar la persona';
            }

          } catch (err) {
            console.error("Error al validar la foto: ", err);
            showError('Error al procesar la validación: ' + err.message);
          }
        }

        function imageToBase64(imgElement) {
          return new Promise((resolve, reject) => {
            const img = new Image();
            img.crossOrigin = 'Anonymous';
            img.src = imgElement.src;
            img.onload = () => {
```

```
                const canvas = document.createElement('canvas');
                canvas.width = img.width;
                canvas.height = img.height;
                const ctx = canvas.getContext('2d');
                ctx.drawImage(img, 0, 0);
                const base64 = canvas.toDataURL('image/jpeg');
                resolve(base64);
            };
            img.onerror = () => reject(new Error("No se pudo cargar la imagen."));
        });
    }

    window.onunload = function() {
        if (stream) {
            stream.getTracks().forEach(track => track.stop());
        }
    };

    window.onload = initCamera;
  </script>
</body>
</html>
```

Anexo 6 Código de Node-RED en JSON


```
[
    {
        "id": "71c20c1e80582018",
        "type": "tab",
        "label": "AWS IoT Core - Relay Control",
        "disabled": false,
        "info": ""
    },
    {
        "id": "88b5fda6b095b2c0",
        "type": "mqtt in",
        "z": "71c20c1e80582018",
        "name": "MQTT Input",
        "topic": "validation/led",
        "qos": "0",
        "datatype": "json",
        "broker": "broker1",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 120,
        "y": 100,
        "wires": [
            [
                "bc2d0dcdb2a47204",
                "8c81b150590bc832"
            ]
        ]
    },
    {
        "id": "bc2d0dcdb2a47204",
        "type": "function",
        "z": "71c20c1e80582018",
        "name": "Check Similarity",
```

```
    "func": "let similarity = parseFloat(msg.payload.similitud);\n\nif (isNaN(similarity)) {\n    node.warn(\"La similitud no es un número válido: \" + msg.payload.similitud);\n    return null;\n}\n\nlet messages = [];\n\n// Relé con trigger bajo: 0 activa, 1 desactiva\nif (similarity >= 95 && similarity <= 100) {\n    // Activar relé (LOW)\n    messages[0] = { payload: 0 };\n    // Desactivar relé tras delay (HIGH)\n    messages[1] = { payload: 1 };\n} else {\n    // Mantener relé apagado\n    messages[0] = { payload: 1 };\n    messages[1] = null;\n}\n\nreturn messages;\n",
    "outputs": 2,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 340,
    "y": 140,
    "wires": [
      [
        "acf95c7b37e8c31f"
      ],
      [
        "40cf722f97144319"
      ]
    ]
  },
  {
    "id": "acf95c7b37e8c31f",
    "type": "rpi-gpio out",
    "z": "71c20c1e80582018",
    "name": "Control Relé GPIO 17",
    "pin": "18",
    "set": true,
    "level": "1",
    "freq": "",
    "out": "out",
    "bcm": true,
    "x": 720,
    "y": 80,
    "wires": []
  },
```

```
{
    "id": "40cf722f97144319",
    "type": "delay",
    "z": "71c20c1e80582018",
    "name": "Delay para apagar relé",
    "pauseType": "delay",
    "timeout": "5",
    "timeoutUnits": "seconds",
    "rate": "1",
    "nbRateUnits": "1",
    "rateUnits": "second",
    "randomFirst": "1",
    "randomLast": "5",
    "randomUnits": "seconds",
    "drop": false,
    "outputs": 1,
    "x": 480,
    "y": 220,
    "wires": [
        [
            "acf95c7b37e8c31f"
        ]
    ]
},
{
    "id": "8c81b150590bc832",
    "type": "debug",
    "z": "71c20c1e80582018",
    "name": "Debug MQTT Payload",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "x": 340,
    "y": 80,
    "wires": []
},
```

```
    {
        "id": "broker1",
        "type": "mqtt-broker",
        "name": "AWS IoT Core",
        "broker": "a1e2y4tm1rmbh8-ats.iot.us-east-2.amazonaws.com",
        "port": "8883",
        "tls": "17c01d25d88ef70f",
        "clientid": "ValidationDevice",
        "autoConnect": true,
        "usetls": true,
        "protocolVersion": 4,
        "keepalive": "60",
        "cleansession": true,
        "autoUnsubscribe": true,
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": "",
        "birthMsg": {},
        "closeTopic": "",
        "closeQos": "0",
        "closePayload": "",
        "closeMsg": {},
        "willTopic": "",
        "willQos": "0",
        "willPayload": "",
        "willMsg": {},
        "userProps": "",
        "sessionExpiry": ""
    },
    {
        "id": "17c01d25d88ef70f",
        "type": "tls-config",
        "name": "",
        "cert": "/home/Carmen/lambda_function/ValidationDevice.cert.pem",
        "key": "/home/Carmen/lambda_function/ValidationDevice.private.key",
        "ca": "/home/Carmen/lambda_function/AmazonRootCA1.pem",
        "certname": "",
        "keyname": "",
        "caname": "",
        "servername": "",
        "verifyservercert": true,
        "alpnprotocol": ""
    }
]
```