FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

IMPLEMENTACIÓN DE UN SISTEMA PARA LA ADMINISTRACIÓN DE LAS FUNCIONES BÁSICAS DE UN CISCO MERAKI MR33 MEDIANTE CISCO DEVNET Y AMAZON ALEXA EN EL DATA CENTER ACADÉMICO DE LA UNIVERSIDAD DE LAS AMÉRICAS.

AUTORES

VÍCTOR ANDRÉS BURGOS BUENAÑO

CRISTIAN DAVID PROAÑO JIMÉNEZ

AÑO

2020

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

IMPLEMENTACIÓN DE UN SISTEMA PARA LA ADMINISTRACIÓN DE LAS FUNCIONES BÁSICAS DE UN CISCO MERAKI MR33 MEDIANTE CISCO DEVNET Y AMAZON ALEXA EN EL DATA CENTER ACADÉMICO DE LA UNIVERSIDAD DE LAS AMÉRICAS.

Trabajo de Titulación presentado en conformidad a los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de Información e Ingeniero en Redes y Telecomunicaciones.

Profesor Guía

Ing. Ángel Jaramillo Alcázar, Mg.

Autores

Víctor Andrés Burgos Buenaño

Cristian David Proaño Jiménez

Año

2020

DECLARACIÓN PROFESOR GUÍA.

“Declaro haber dirigido el trabajo, Implementación de un sistema para la administración de las funciones básicas de un Cisco Meraki MR33 mediante Cisco DevNet y Amazon Alexa en el data center académico de la Universidad de las Américas, a través de reuniones periódicas con los estudiantes Víctor Andrés Burgos Buenaño y Cristian David Proaño Jiménez, en el semestre 202020, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”.

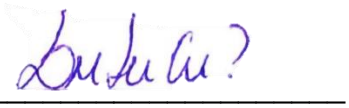
Ángel Gabriel Jaramillo Alcázar.

Magister en Gerencia de Sistemas y Tecnologías de la Información

C.I:1715891964

DECLARACIÓN PROFESOR CORRECTOR.

“Declaro haber dirigido el trabajo, Implementación de un sistema para la administración de las funciones básicas de un Cisco Meraki MR33 mediante Cisco DevNet y Amazon Alexa en el data center académico de la Universidad de las Américas, a través de reuniones periódicas con los estudiantes Víctor Andrés Burgos Buenaño y Cristian David Proaño Jiménez, en el semestre 202020, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”.



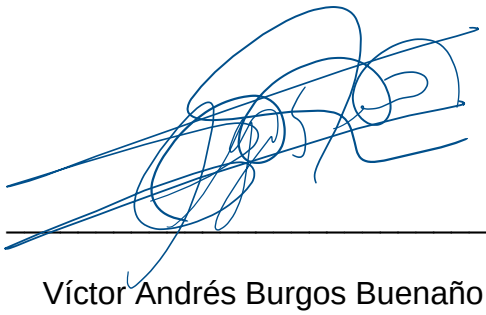
Luis Santiago Criollo Caizaguano

Máster en Redes de Comunicación

CI: 1717112955

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE.

“Declaro que este trabajo es original, de nuestra autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes”.



Víctor Andrés Burgos Buenaño

CI: 1723962807

Cristian David Proaño Jiménez

CI: 1103577449

Agradecimiento

Primero quisiera darle gracias a Dios por derramar bendiciones a lo largo de mi vida. Darle gracias a toda mi familia por el apoyo que me brindan y a superarme logrando cada objetivo propuesto en mí vida. A mis maestros por la guía y las enseñanzas que me han brindado a lo largo de mi carrera. Al Ing. Ángel Jaramillo por la ayuda en la realización del proyecto. A mis amigos de la infancia y en especial a mis amigos Vikingos por todo lo que me ayudaron y disfruté día a día junto a ellos a lo largo de mi etapa universitaria.

Agradecimiento

A mis padres Fernando y Silvana por siempre velar por mi bienestar, gracias a sus consejos y confianza he podido alcanzar una meta más. A mis hermanos por su apoyo incondicional, siempre serán mi ejemplo a seguir. A mis maestros por brindarme todos sus conocimientos a lo largo de este camino. Al Ing. Ángel Jaramillo por su apoyo y tiempo brindado. A mis primos y amigos que siempre han confiado en mí, me han apoyado y motivado para nunca decaer y siempre seguir adelante.

Dedicatoria

Este trabajo es dedicado a toda mi familia, a mis padres, a todos los que me han brindado su apoyo, en especial a mi padre Víctor Burgos Vallejo que gracias al esfuerzo y sacrificio de él logré estudiar y culminar mi carrera y a mi hermana Karen Burgos Buenaño por la ayuda que me ha brindado y el empuje que me permitió cumplir este objetivo.

Dedicatoria

Con todo cariño dedico este trabajo a mis familiares, amigos y a todas las personas que me han ayudado a que esto sea posible. A mis padres que siempre me han forjado con grandes valores para seguir adelante. A mis hermanos que con su confianza, presión y apoyo me han motivado a lo largo de este camino.

En memoria a mi madre, Silvana Jiménez, por ser ejemplo de superación, lealtad, fe y amor.

RESUMEN

En la actualidad cada día se presentan nuevos avances tecnológicos, por lo cual mediante este trabajo de titulación se pretende innovar y mejorar la manera en cómo se administra una red mediante Routers AP.

Este proyecto tiene como objetivo facilitar a los administradores de red el manejo de un data center, además de estar enfocado en la inclusión de personas con discapacidad, ya que hoy en día las empresas con un numero de veinticinco o más trabajadores tiene la obligación de contratar a una persona con discapacidad. Por lo tanto, por medio de esta implementación se busca beneficiar a toda persona que desee trabajar en esta área.

Para el desarrollo, se ha utilizado un *Access Point* de la marca Cisco, modelo Meraki MR33 que permite la interacción con diferentes plataformas para su administración para finamente poder ser controlado por el usuario mediante un asistente de voz.

La implementación del proyecto será instalada en el data center Académico de la Universidad de las Américas. Por consiguiente, la universidad contará con dos proyectos con un mismo enfoque, donde se podrá gestionar por comandos de voz tanto dispositivos Meraki como también el UCS.

Lo que se espera a futuro es la implementación e incorporación de nuevos proyectos relacionados con la misma temática (gestión por voz), ya que se facilitaría la administración y optimizaría el tiempo de manejo y se lograría tener toda una red unificada y controlada por voz.

Palabras Claves: Networking, Meraki, asistente virtual.

ABSTRACT

Currently, new technological advances are presented every day, which is why this degree work aims to innovate and improve the way in which a network is managed using AP Routers.

This project aims to facilitate network administrators the management of a data center, in addition to being focused on the inclusion of people with disabilities, since today companies with a number of twenty-five or more workers have the obligation to hire to a person with a disability. Therefore, through this implementation, we seek to benefit everyone who wishes to work in this area.

For the development, an Access Point of the Cisco brand has been used, Meraki MR33 model that allows interaction with different platforms for its administration to be finely controlled by the user through a voice assistant.

The implementation of the project will be installed in the Academic data center of the University of the Americas. Consequently, the university will have two projects with the same focus, where both Meraki devices and the UCS can be managed by voice commands.

What is expected in the future is the implementation and incorporation of new projects related to the same theme (voice management), since administration would be facilitated, and management time would be optimized and a whole unified and voice-controlled network would be achieved.

Keywords: Networking, Meraki, virtual assistant.

ÍNDICE

1. Capítulo I. Introducción 1

1.1. Objetivo General 2

1.2. Objetivos Específicos 2

1.3. Alcance 3

1.4. Justificación 3

2. Capitulo II. Marco Teórico 4

2.1. Cisco Meraki 4

2.1.1. Meraki MR33 5

2.1.2. Meraki Developer Dashboard 6

2.2. Cisco DevNet 6

2.2.1. Sandbox DevNet 6

2.3. Asistente Virtual 7

2.4. Windows PowerShell 8

2.5. Amazon Web Services 8

2.5.1. Amazon Lambda 9

2.5.2. Alexa Web Information Service (AWIS) 9

2.5.3. Amazon S3 9

2.6. Amazon Echo 11

2.6.1. Amazon Echo DOT 3ra generación 11

2.6.2. Alexa Cloud 11

2.6.3. Amazon Skill Server 12

2.6.4. Aplicación Alexa 12
2.7. Arquitectura Amazon Alexa 12
2.8. Postman 13
2.9. Python 3.6 14
3. Capitulo III. Desarrollo e Implementación 14
3.1. Diseño e Integración de Plataformas 14
3.2. Cisco DevNet SandBox 15
3.3. Dashboard Meraki 18
3.4. Creación y configuración de plataforma Amazon Web Services 19
3.4.1. Amazon Web Services 19
3.4.2. Creación de Rol 20
3.4.3. Creación de Función en Lambda 24
3.5. Configuración e Integración de Alexa con AWS 28
3.6. Creación de código Python 33
4. Capitulo IV. Migración del sistema para la administración del UCS al lenguaje de programación Python 3.7 36
5. Capitulo V. Pruebas y Resultados 42
6. CONCLUSIONES Y RECOMENDACIONES 49
6.1. Conclusiones 49
6.2. Recomendaciones 50

REFERENCIAS..... 52

1. Capítulo I. Introducción

Conforme avanza el tiempo, los avances tecnológicos tienen un mayor enfoque en la innovación y automatización, brindando una mejora continua en la administración y eficiencia de las redes. La plataforma para desarrolladores Cisco DevNet ofrece la posibilidad de tener redes más adaptables en cuanto a su infraestructura.

DevNet es una comunidad creada en el 2014 por la empresa Cisco, esta plataforma nació con el propósito de ayudar y fomentar la creación y desarrollo de nuevos proyectos, brindando soluciones innovadoras. ("Cisco DevNet: APIs, SDKs, Sandbox, and Community for Cisco Developers", 2020).

La plataforma de Cisco DevNet ofrece diferentes alternativas para los desarrolladores y usuarios como IoT, cloud, software, seguridad, centro de datos, y desarrollo de código abierto, herramientas que serán el futuro para la aplicación de nuevas tecnologías.

Para realizar la administración y configuración de un data center, es necesario ingresar a diferentes interfaces, a más de ejecutar funciones por medio de líneas de comando. Debido a esto, se resolvió por implementar un sistema para la administración de las funciones básicas que tiene un Cisco Meraki MR33 mediante Cisco DevNet, servicios web de Amazon y Amazon Alexa, herramientas que permiten realizar las configuraciones por medio de comandos de voz. Además, de complementar un proyecto realizado anteriormente en la carrera y en el cual el enfoque era la administración de un UCS.

“Meraki ofrece servicios Wifi, enrutamiento y conmutación controlados en la nube, así como acceso programático a través de una API REST.” ("Cisco DevNet Learning Labs", 2020).

“La API de Meraki Dashboard permite a los desarrolladores manejar tareas tediosas de forma rápida y sencilla. La API utiliza convenciones REST familiares sobre HTTP con datos JSON.” ("Cisco Meraki - Create with the Meraki Platform", 2020).

“Alexa Skills Kit (ASK) es un conjunto de herramientas, documentación, muestras de código y API en self-service con el que se puede añadir Skills a Alexa de forma rápida y sencilla.” ("Developer Amazon", 2020).

“Amazon Echo es un dispositivo que hace uso de Alexa, proporcionando un servicio de voz basado en la nube (conocido como servicio cloud) que, gracias a la tecnología de inteligencia artificial, aprenden, siendo capaces de contestar preguntas, ofrecer resultados, noticias, reproducir música y por supuesto, controlar distintos dispositivos.” (Gallardo Vázquez, 2019).

“Python SDK permite una fácil integración con los procesos y herramientas de administración de TI existentes. El SDK de Python le permite consultar, crear, modificar y eliminar los Objetos administrados.” ("Cisco DevNet: APIs, SDKs, Sandbox, and Community for Cisco Developers", 2020)

1.1. Objetivo General

Implementar un sistema para la administración de las funciones básicas de un Cisco Meraki MR33 mediante Cisco DevNet y Amazon Alexa en el data center académico de la Universidad de las Américas.

1.2. Objetivos Específicos

- Analizar las funcionalidades básicas del Cisco Meraki MR33 y los servicios e infraestructura necesarios para el desarrollo del proyecto.
- Implementar el sistema considerando las funcionalidades dimensionadas para el proyecto.
- Realizar la migración del sistema para la administración del UCS al lenguaje de programación Python 3.7

1.3. Alcance

El alcance del siguiente proyecto de titulación es la implementación de un sistema para la administración de las funciones básicas de un Cisco Meraki MR33 por medio de la plataforma Cisco DevNet y Amazon Alexa en el data center de la Universidad de las Américas. Además de complementar y migrar a Python 3.7 el sistema para la administración de una red realizado anteriormente.

La finalidad del siguiente trabajo de titulación es la administración de las redes mediante la utilización de herramientas como Alexa *Skills Kit*, las cuales se pretenden utilizar en el data center ubicado en las instalaciones de la Universidad de las Américas.

La parte de programación será elaborada mediante el lenguaje Python, el cual permite elaborar diferentes codificaciones y nuevas funciones como habilitar y deshabilitar una red, visualizar el inventario de las redes existentes, ver los terminales conectados, entre otros. Estos comandos se vinculan al AWS (Servicios Web de Amazon) y se usará el dispositivo de control de voz Amazon Echo, para reconocer y sincronizar los comandos haciendo posible el poder administrar Cisco Access Point Meraki de una forma eficaz.

1.4. Justificación

Debido al incremento masivo de avances tecnológicos cada vez se busca que los sistemas sean más eficientes y sobre todo autónomos de esta manera se brinda facilidad a los usuarios al momento de administrar y realizar diversas configuraciones, en base a estas razones se pretende con este trabajo de titulación lograr seguir innovando y avanzando a la par con la tecnología.

El sistema busca integrar e incluir a personas con discapacidades a que tengan la oportunidad de laborar de una forma normal en el ambiente tecnológico, ya que este proyecto está diseñado para brindar comodidad y accesibilidad a toda persona. De igual forma se logrará que futuros proyectos utilicen asistentes virtuales para realizar funciones como seguridad de redes, centro de datos, cloud, entre otros.

Con la incorporación de este proyecto los administradores de redes tendrán mayores facilidades para gestionar el data center de la Universidad de las Américas. A su vez, se está optimizando tiempos ya que no se debe ejecutar comandos por medio de consola o interfaz gráfica.

Adicionalmente, tenemos que con la migración al lenguaje de programación Python 3.7 del sistema para la gestión del UCS se logra que los equipos tengan soporte técnico en caso de que exista la presencia de fallas. De esta manera se logra que el data center sea más eficiente, moderno e intuitivo para el administrador.

2. Capitulo II. Marco Teórico

2.1. Cisco Meraki

Es un dispositivo de acceso a internet (Access Point AP), el cual tiene el propósito de expandir la señal del internet de los terminales que se encuentran lejos del dispositivo enrutador. El modelo fue creado por la empresa de Cisco, quienes desarrollan distintos modelos, los cuales se puede encontrar en la página oficial de Cisco.

Algunos dispositivos que tienen características similares a las de Cisco Meraki MR33, son los siguientes:

- MR20
- MR30H
- MR36
- MR42
- MR42E
- MR45
- MR46
- MR52
- MR53

- MR53E
- MR55
- MR56

2.1.1. Meraki MR33

Dispositivo AP que posee características mejoradas con respecto a los dispositivos mencionados en la tabla 1, como la gestión centralizada desde la nube, distintos niveles de seguridad y con un sistema autoconfigurable. Cuenta con características como:

- Dimensión

Posee unas dimensiones de 215 mm * 110 mm * 32 mm.

- Antenas

Este dispositivo posee antenas omnidireccionales dual band, trabajan con una ganancia de 3.8 dBi en la banda de 2.4 GHz y con una ganancia de 3.9 con la banda de 5GHz. Además, hay que mencionar que este tipo de antenas tiene el sistema de MIMO 2x2 que permite actualizar la red del cliente por medio de la tecnología quad-radio.

- Nube

Mediante el equipo de cisco Meraki y la suscripción a la plataforma se tiene la gestión de los equipos remotamente desde la nube, además de obtener informes de manera automática.

- Seguridad

Este dispositivo posee herramientas de seguridad innovadoras, debido a que tiene características como:

- Cortafuegos para invitados.
- Posee un escáner de antivirus, el cual se encuentra incorporado para el análisis y detección de software malicioso, así como posibles atacantes.

- Con la gestión de este dispositivo se permite la creación de reglas de firewall específicas.
- El firmware se mantiene actualizado desde la nube.
- Cuenta con alertas las 24 horas al día, en tiempo real.

("Meraki MR33 | Access Point | 802.11 AC Wireless", 2020).

2.1.2. Meraki Developer Dashboard

Esta plataforma es interactuar con el dispositivo, aquí se obtiene los datos de cada dispositivo.

2.2. Cisco DevNet

Plataforma tecnológica lanzada en el 2014 que ofrece diversas herramientas para el desarrollo e implementación de proyectos, además de contar con otros recursos como código abierto, laboratorios, *API's* entre otras herramientas que permiten crear y realizar nuevas soluciones innovadoras para ser implementadas en el campo de las redes.

2.2.1. Sandbox DevNet

Es una plataforma realizada por Cisco, que facilita con laboratorios tecnológicos para el desarrollo de proyectos en distintas categorías, las cuales son:

- Networking
- IoT
- Data Center
- Collaboration
- Cloud
- Analytics and Automation SW
- Security

Esta plataforma es de gran utilidad ya que se pueda acceder desde cualquier parte en el mundo, además de contar con varios repositorios los cuales ayudan para el desarrollo e implementaciones de nuevos proyectos.

El uso de la plataforma bajo reserva cuenta con diversas características las cuales poseen sus pros y contras:

Pros

- Se puede reservar la plataforma como máximo de siete días.
- Tienen un ambiente privado para el desarrollo de pruebas.
- La plataforma contiene instrumentos en el laboratorio automatizados.

Contras

- Requiere una reserva para trabajar en la plataforma.
- Se necesita tener una aplicación para la conexión de la VPN.
- Para el trabajo en la plataforma se necesita una configuración de al menos una hora. ("Cisco DevNet: APIs, SDKs, Sandbox, and Community for Cisco Developers", 2020).

2.3. Asistente Virtual

Un asistente virtual es un ordenador el cual brinda la facilidad de cumplir con diferentes tareas, ya sean de búsqueda o de acciones, las cuales están dentro del mundo tecnológico, como por ejemplo encender luces o apagar dispositivos en el hogar.

El asistente virtual lo debemos a John McDonough, ya que mediante su aporte realizado a la comunidad de Cisco y con la combinación de plataformas como Amazon Web Services y DevNet. Se puede tener el control y la gestión por comandos de voz, en este caso de dispositivos Meraki de la línea de Cisco.

En la actualidad se tiene algunos asistentes de voz como el de Siri perteneciente a Apple, Cortana de Google, Alexa de Amazon, Watson de IBM entre otros. Pero se tiene que tomar en cuenta que la integración de estos dispositivos se ha dado más

en proyectos relacionados con domótica, y no han sido utilizados en proyectos relacionados con el área de *Networking*. Debido a sus políticas, ya que restringen toda conexión o integración de complementos o sencillamente no han requerido realizar el esfuerzo de integrar esos dispositivos.

Hoy en día disponemos de varios asistentes virtuales, pero no se les ha dado la debida importancia del uso que se le puede dar, ya que por medio de esta herramienta se puede desarrollar diferentes proyectos innovadores que integren a personas con discapacidad. Es por eso por lo que la empresa de tecnología de Cisco busca implementar toda clase de proyectos relacionados con la innovación diaria.

Otro proyecto relacionado con asistentes virtuales e integración de otras plataformas fue desarrollado en la Universidad de las Américas, el cual consiste en la creación de Vlans mediante comandos de voz utilizando el dispositivo Amazon Echo.

2.4. Windows PowerShell

“Powershell es un lenguaje de script y un entorno creado para la automatización de tareas por Microsoft. Microsoft desarrolló Powershell por la deficiencia de los otros lenguajes de administración de sistemas operativos, como por ejemplo Scripts, que es muy limitado, o el VBScript que no es tan rápido. Powershell fue lanzado en 2006 para Windows 7 and Server 2008R2, aunque se puede encontrar para otras versiones de Windows.” (Aranda, 2014).

2.5. Amazon Web Services

Es una plataforma que tiene como propósito trabajar por medio de la nube, ofreciendo distintos servicios remotos como almacenamiento de forma segura de bases de datos, además de albergar páginas web dinámicas, entregar archivos de forma rápida a cualquier parte del mundo por medio de la red. También Amazon ofrece servicios de Alexa y Lambda, esos servicios son los que se utiliza en el desarrollo del proyecto de titulación.

2.5.1. Amazon Lambda

“AWS Lambda le permite ejecutar código sin aprovisionar ni administrar servidores. Puede ejecutar código para prácticamente cualquier tipo de aplicación o servicio de *back-end,* todo con cero administraciones. Simplemente cargue su código y Lambda se encarga de todo lo necesario para ejecutar y escalar su código con alta disponibilidad.” ("AWS Lambda – Serverless Compute - Amazon Web Services", 2020).

El servicio que ofrece Amazon Lambda es muy distinto al que tienen las otras plataformas, debido que sirven siempre y cuando las aplicaciones en las que se vayan a utilizar permitan la escritura de lenguajes que hayan sido aceptados o aprobados por la plataforma de Amazon Web Services Lambda.

2.5.2. Alexa Web Information Service (AWIS)

“AWIS ofrece una plataforma donde se pueden crear soluciones y servicios web innovadores basados en la gran cantidad de información que tiene Alexa sobre sitios web, accesible con una API de servicios web.” ("Alexa Web Information Service", 2019).

Otro servicio es Alexa para negocios que tiene herramientas necesarias para gestionar dispositivos compatibles con Alexa, registro de usuarios y asignación de habilidades. Además, permite crear sus propias habilidades de voz mediante la *API* de Alexa para negocios. (¿” What Is Alexa for Business? - Alexa for Business", 2019).

2.5.3. Amazon S3

El servicio que brinda Amazon S3 consiste en una prestación de almacenamiento, el cual otorga cualidades como disponibilidad de datos, escalamiento y la más importante seguridad hacia los clientes. Es por eso por lo que pueden realizar el almacenamiento de datos y la protección de esta. Los más comunes son “sitios web, aplicaciones móviles, procesos de copia de seguridad y restauración, operaciones

de archivado, aplicaciones empresariales, dispositivos IoT y análisis de big data.” ("AWS | Almacenamiento de datos seguro en la nube (S3)", 2020).

Amazon S3 tiene la peculiaridad de uso fácil y administración, permitiendo cumplir satisfactoriamente con los requerimientos empresariales por medio de la organización de datos y la configuración de los accesos. Este servicio fue diseñado con el propósito de brindar durabilidad del 99.9% (11 nueves), ya que este servicio alberga aplicaciones y millones de datos de empresas de alrededor del mundo.

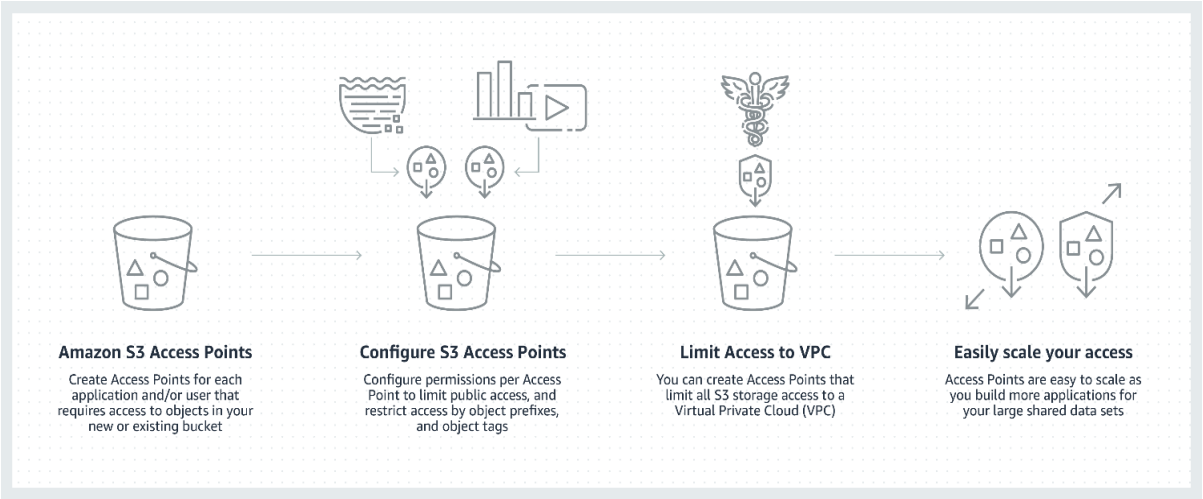


Figura 1. Puntos de acceso S3.

Tomado de: (“https://aws.amazon.com/es/s3/”, 2020).

El servicio ofrecido por medio de Amazon S3 es uno de los más completos debido a que los complementos ofertados son la disponibilidad, rendimiento, durabilidad, escalabilidad, una diversidad de clases de almacenamiento rentables, seguridad, administración de datos y almacenamiento en la nube con el debido soporte técnico.

En la imagen vista de los puntos de acceso, se muestra la simplificación que se lleva para administrar el acceso a los datos que se utilizan para las aplicaciones. Los puntos de acceso tienen como propósito dar “una ruta personalizada hacia un bucket”, el cual tiene por defecto un nombre de host y una política permisos de accesos que será realizada por medio del punto de acceso.

2.6. Amazon Echo

EL dispositivo tiene la funcionalidad de un asistente de voz para los usuarios. El equipo empieza a funcionar cuando el usuario menciona la palabra por el comando de voz “Alexa”, una vez encendido se transmite el comando mencionado al servicio que también ofrece Amazon, el cual es Amazon Alexa *Cloud*, finalmente el comando de voz se transmite para el dispositivo y se reproduce la tarea en Amazon Echo.

2.6.1. Amazon Echo DOT 3ra generación

Es un parlante (altavoz) inteligente que tiene incorporado Alexa, asistente virtual, que permite realizar las funciones que se le ordene como, por ejemplo

- Reproducción de música.
- Controlar tus dispositivos *Smart* de tu hogar.
- Llamar a contactos por comandos de voz.
- Obtener información (Clima, Noticias, Tráfico)
- Recordatorios

El dispositivo Echo Dot de 3ra generación tiene la característica de trabajar con conectividad WIFI en dos bandas (2.4 GHz y 5 GHz) con el protocolo de red 802.11 a/b/g/n. Además de poder trabajar con conectividad Bluetooth. Pero una de las cosas que hacen que este dispositivo no sea completo es la compatibilidad con el sistema Mac OS X por temas de permisos. Mencionar, además que necesita un complemento para sistemas operativos de Fire Os, Android y el complemento es el aplicativo para el funcionamiento y conectividad entre ambos.

2.6.2. Alexa Cloud

Este servicio se encarga de manejar el reconocimiento de voz, después asigna correctamente el comando de voz mencionado. Los comandos se distinguen como características especiales de Alexa, las cuales están almacenadas en los servidores, esto siendo operado por la empresa de Amazon. Una vez recibido y

almacenado en la nube del servidor de Alexa, este es devuelto al dispositivo de Echo de Amazon.

2.6.3. Amazon Skill Server

También conocidos como habilidades, los *skills* de Amazon son trabajos que pueden ser realizados por Alexa. Estos requerimientos son hechos por los usuarios y deben estar previamente configurados en los servidores de Alexa. Una de las características principales es el acceso a las configuraciones ya que se encuentran almacenadas en la nube y los usuarios pueden acceder a todos los *skills* por medio de dos formas y estas son:

- Aplicativo de Amazon Alexa
- Página Web de Amazon Alexa

2.6.4. Aplicación Alexa

La aplicación de Amazon es un software móvil, la cual se encarga de la administración de distintos servicios, algo que es muy práctico por su fácil acceso a la plataforma, donde los usuarios pueden realizar sus configuraciones y en este caso las modificaciones de los *skills*, para la administración de los dispositivos de Amazon.

Al interactuar los usuarios con este aplicativo y sus artilugios, el servicio de la nube de Alexa se encarga de enviar documentos con informes al aplicativo acerca de lo que se ha realizado. Los informes son muy importantes debido a que la información que se obtuvo es adjuntada con la grabación de la pregunta que se realizó por el usuario en el dispositivo Echo de Amazon.

2.7. Arquitectura Amazon Alexa

En la arquitectura de Alexa se puede encontrar distintos servicios, pero como se había mencionado anteriormente aquí se concentrará en el asistente de voz y los usuarios pueden acceder por sistemas integrados a los sistemas de Amazon Alexa.

En la imagen 2 se detalla los componentes que tiene la arquitectura de Amazon Alexa.

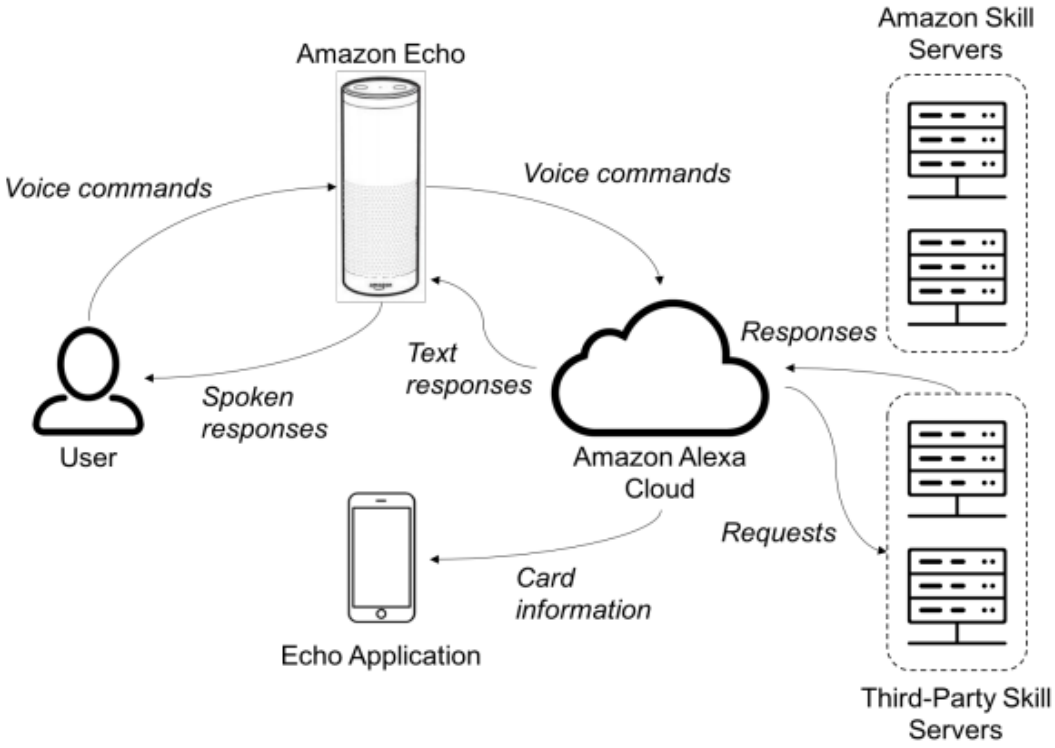


Figura 2. Arquitectura de Amazon Alexa y Amazon Echo.

Tomado de: (Haack, Severance, Wallace & Wohlwend, 2017).

2.8. Postman

Postman es una plataforma que ayuda a los desarrolladores para la creación, compartición y pruebas que se realizan con las *API's* de cada programa. Esta herramienta es muy buena ya que permite que el trabajo sea más ágil y eficiente.

Una de las ventajas de trabajar con este programa es que se adapta a cualquier entorno, ya sea en desarrollo, pruebas, control de calidad, entre otras. ("Postman", 2020).

En específico el programa brinda su servicio con la obtención de datos del dispositivo Meraki, datos que son usados en el código de Python para realizar la correcta integración de las plataformas.

2.9. Python 3.6

Python es un lenguaje de programación, que se usa generalmente para el diseño y desarrollo de software.

“Python es un lenguaje de programación de alto nivel que se caracteriza por el hecho de ser un lenguaje simple, fácil de leer, escribir y depurar, y además es portable.” (Sarasa, 2017).

Las características que se tienen con este software son los siguientes:

- “Python es un lenguaje muy expresivo, es decir, los programas Python son muy compactos: un programa Python suele ser bastante más corto que su equivalente en lenguajes como C. (Python llega a ser considerado por muchos un lenguaje de programación de muy alto nivel).”
- “Python es muy legible. La sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta más fácil que si utilizáramos otros lenguajes de programación. “
- “Python ofrece un entorno interactivo que facilita la realización de pruebas y ayuda a despejar dudas acerca de ciertas características del lenguaje. “
- “El entorno de ejecución de Python detecta muchos de los errores de programación que escapan al control de los compiladores y proporciona información muy rica para detectarlos y corregirlos. “
- “Python puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objetos. Posee un rico juego de estructuras de datos que se pueden manipular de modo sencillo.” (Marzal et al., 2014).

3. Capitulo III. Desarrollo e Implementación

3.1. Diseño e Integración de Plataformas

Las plataformas principales para realizar la correcta implementación son Cisco Meraki, Servicios Web de Amazon (AWS) y Amazon Alexa.

En el *dashboard* de la plataforma Cisco Meraki se obtienen todos los datos del dispositivo que serán incorporados en el código de Python, de igual forma el *dashboard* permite generar el API Key que sirve para integrar con AWS.

En la plataforma de AWS se debe realizar la creación del rol agregando los permisos de S3 y Amazon CloudWatch Logs, añadir el rol a la función de Lambda previamente creada. Posteriormente cargar el código de Python y realizar las modificaciones necesarias. Añadir el desencadenador Alexa *Skills Kit* y copiar el *ARN* que brinda la función.

En la consola de desarrollo de Alexa, se crean las habilidades que serán ejecutadas por el dispositivo Meraki MR33.

Si los pasos mencionados son efectuados correctamente, se puede ordenar por medio de comandos de voz al dispositivo Amazon Echo Dot que realice las configuraciones del Meraki.

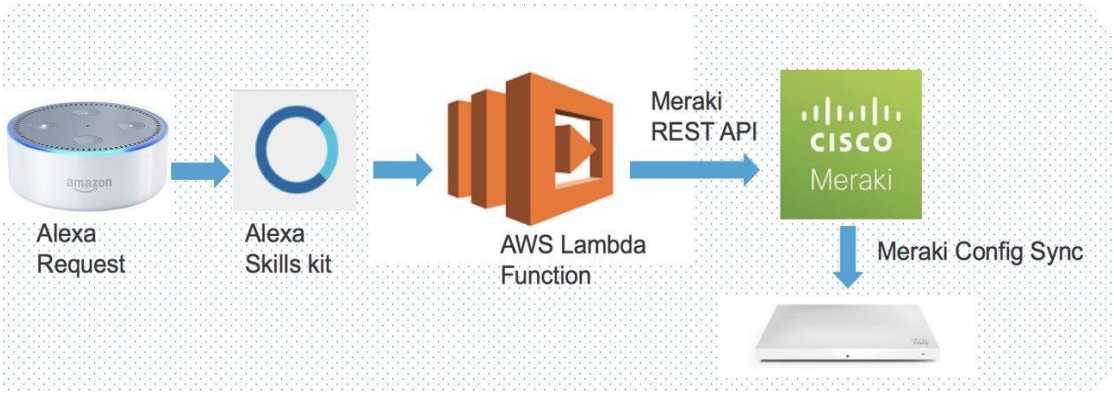


Figura 3. Arquitectura.

Tomado de: (“https://developer.cisco.com/learning/lab/Meraki_Alexa/step/1”, 2020).

3.2. Cisco DevNet SandBox

Para poder utilizar la plataforma de SandBox, donde se realiza las pruebas de simulación de equipos Meraki. Es necesario ingresar al siguiente enlace

https://developer.cisco.com/site/sandbox/. Posteriormente registrase o ingresar con alguna de las cuentas mostradas en la figura 5.

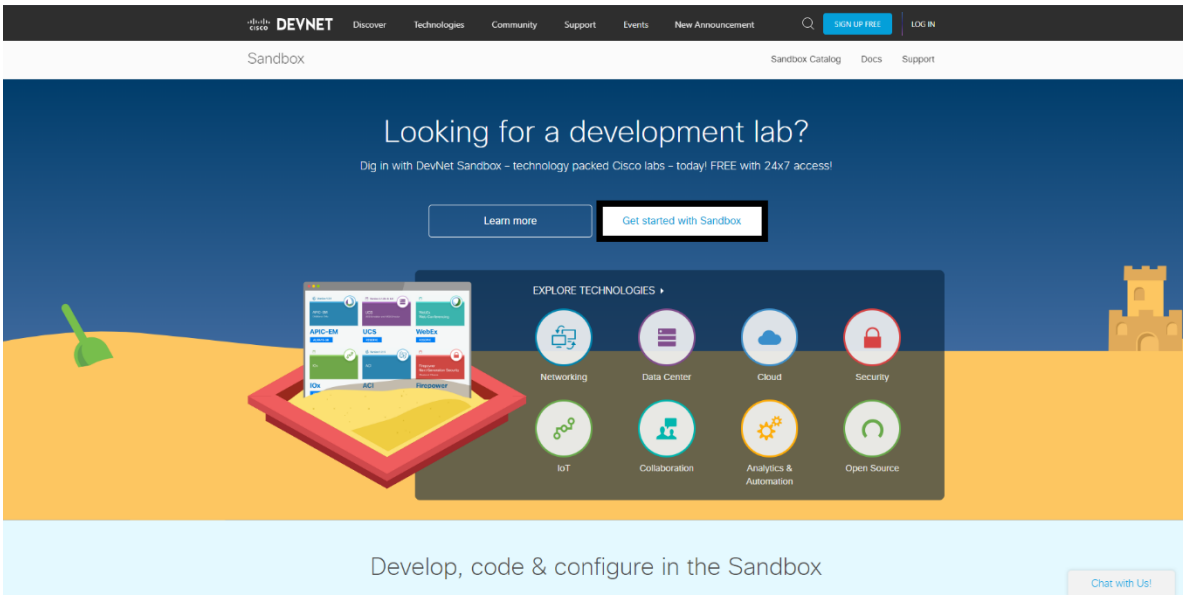


Figura 4. Ingreso a Plataforma *Cisco DevNet SandBox.*

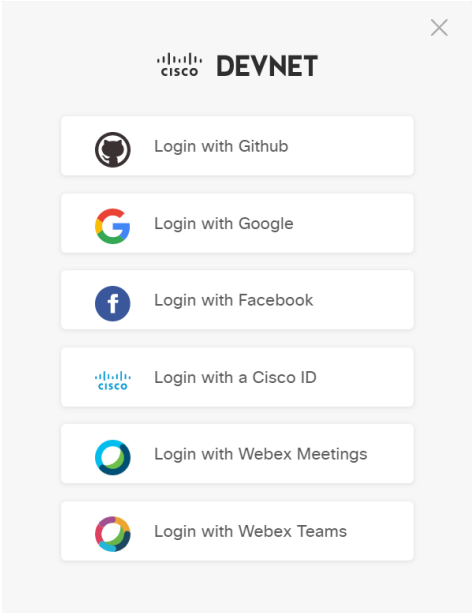


Figura 5. Cuentas disponibles para el registro.

Una vez creada la cuenta se puede acceder a todos los servicios que ofrece DevNet Sandbox. Como se observa en la figura 6, se puede seleccionar y reservar entre tres diferentes dispositivos Meraki (*Always On, Enterprise, Small Business*).

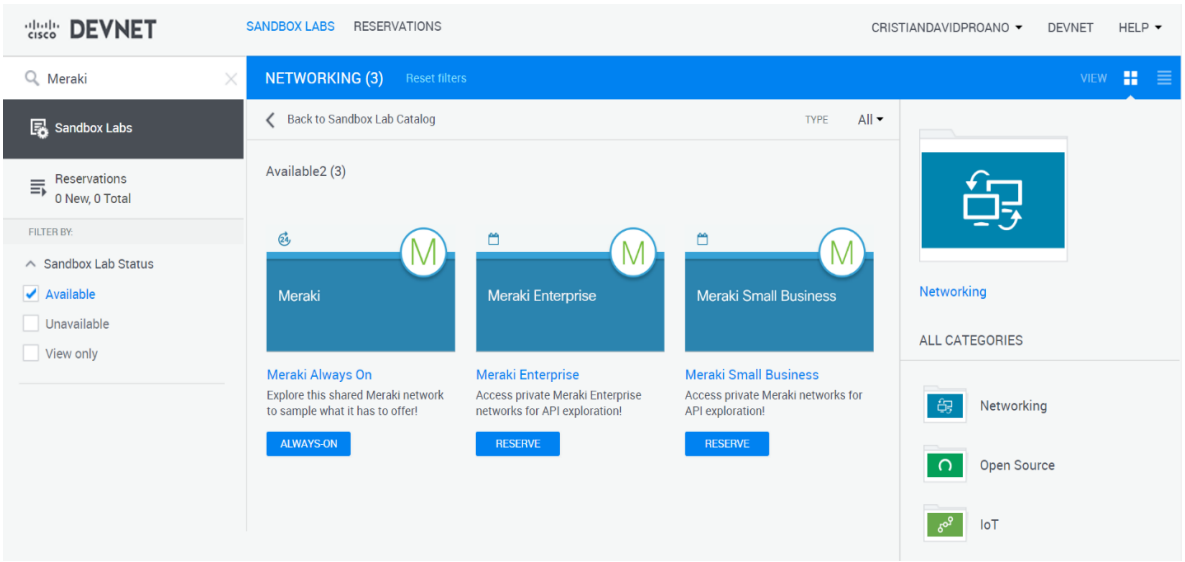


Figura 6. Ingreso a simulador de Meraki.

Cuando se tenga reservado el Meraki se debe esperar 1 minuto para que se realicen las configuraciones. Una vez terminadas se mostrará en la esquina superior derecha que ya se encuentra activo el Meraki y listo para usarse.

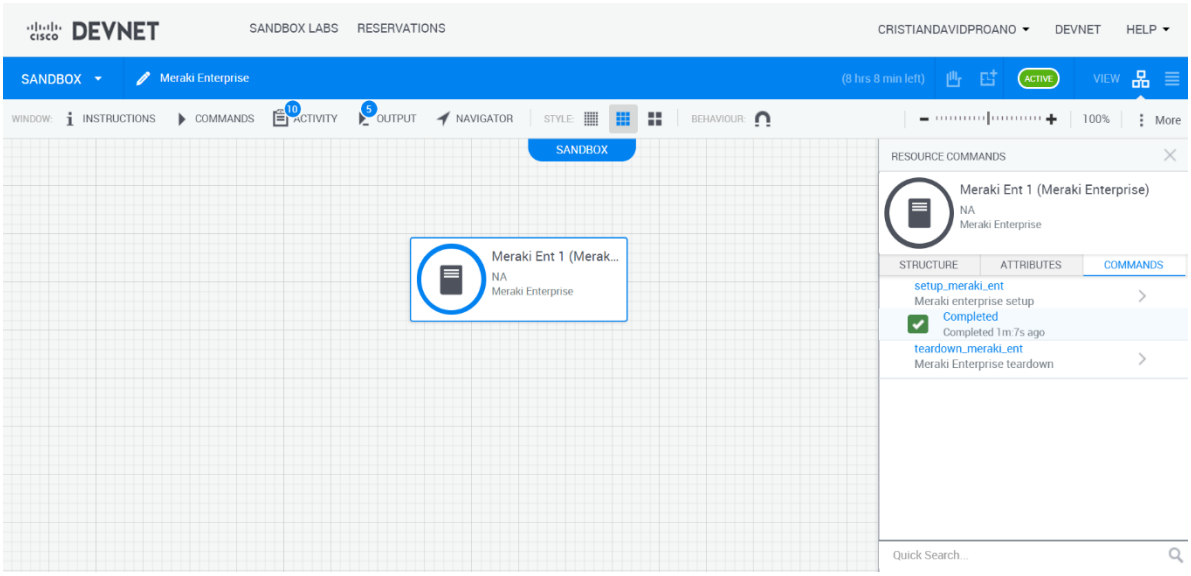


Figura 7. Simulador de equipo Meraki.

3.3. Dashboard Meraki

Una vez que el dispositivo Meraki este activo, se envía un correo a la cuenta vinculada, dando la contraseña y los pasos para acceder al *dashboard* del Meraki.

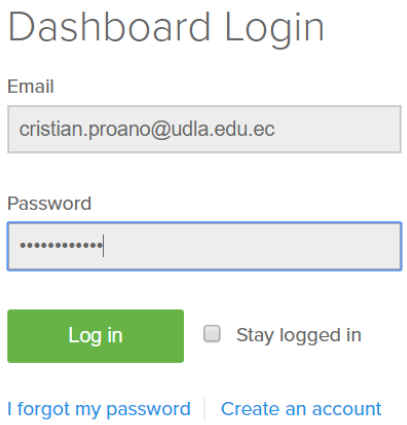


Figura 8. Inicio de sesión a DashBoard Meraki.

En la figura 9 se observa la interfaz del *dashboard* de Meraki, donde se puede realizar diferentes configuraciones, como el nombre de la red, nombre de la organizacion, ver la topología y el estado de la red, entre otras configuraciones.

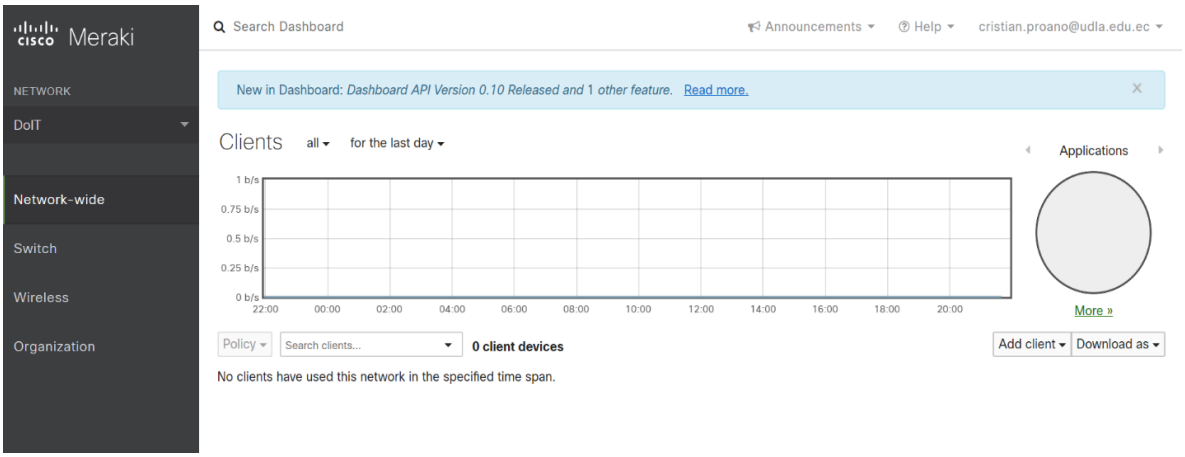


Figura 9. DashBoard de Meraki.

Para obtener el *API Key* se debe dar click en la parte superior derecha donde se encuentra la opción de “Mi Perfil”. Una vez ingresado se debe “Generar una nueva

clave” como se muestra en la figura 10. Esta clave se la debe guardar para usarla en el código de Python.

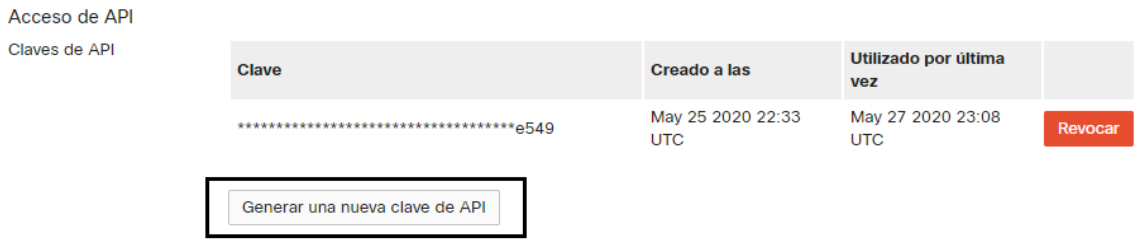


Figura 10. Generador de Claves API.

3.4. Creación y configuración de plataforma Amazon Web Services

3.4.1. Amazon Web Services

Para iniciar en la plataforma de AWS, se debe ingresar al siguiente enlace https://aws.amazon.com/es/console/ donde hay que crear un usuario llenando los datos que se muestran en la figura 11. Si se crea una cuenta gratuita se debe esperar 24 horas para la activación.

Figura 11. Creación de cuenta para la plataforma AWS.

Una vez activa la cuenta, se puede iniciar sesión ingresando las credenciales solicitadas. De esta manera se puede empezar a usar todos los servicios que ofrece Amazon Web Services.

Figura 12. Ingreso a la plataforma AWS.

3.4.2. Creación de Rol

Para la creación de un nuevo rol hay que dirigirse a la parte superior de la página y en la pestaña de Servicios seleccionar “IAM”.

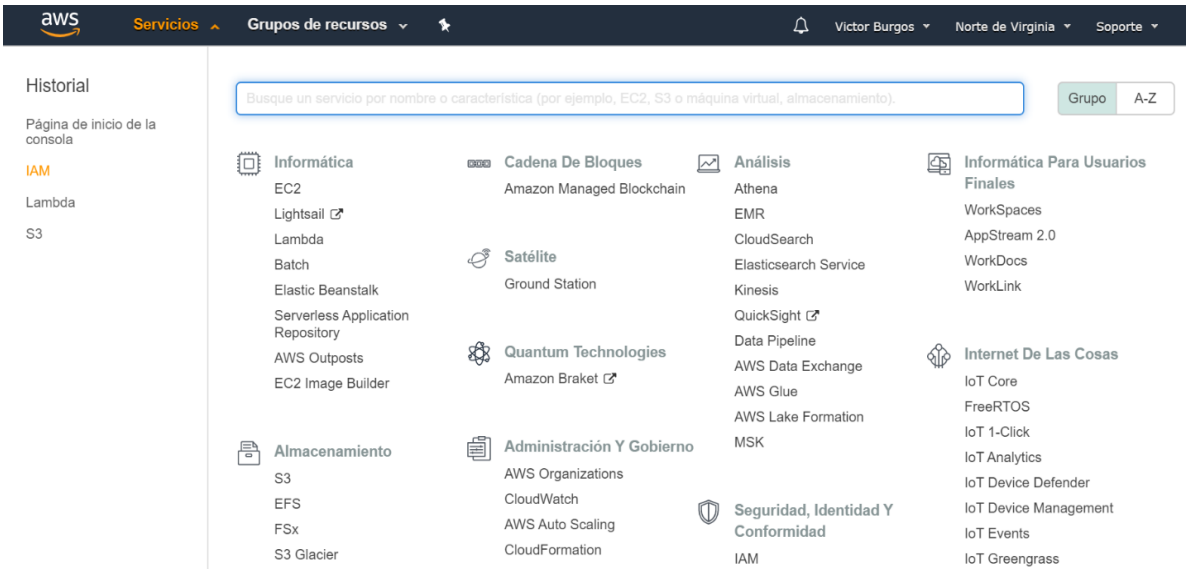


Figura 13. Selección de Servicio.

En la parte izquierda de la página en la opción de Administración del acceso se selecciona Roles, inmediatamente dar click en Crear un rol.

Figura 14. Creación de Rol.

Cuando se crea un rol se debe seguir 4 pasos de configuración. El primero de ellos es seleccionar el caso de uso con el que se desea trabajar, para este proyecto de titulación se escogió Lambda.

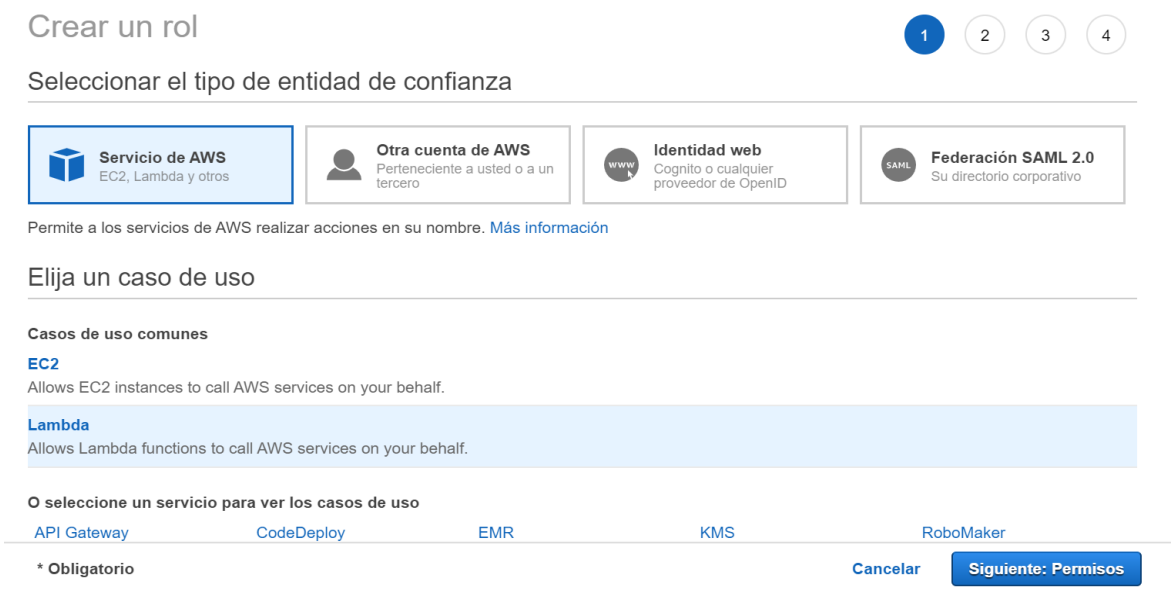


Figura 15. Creación de Rol, selección de uso Lambda.

El segundo paso es escoger que permisos se le otorga al rol. Para ello se puede escribir en el buscador “lambda” y seleccionar “*AWSLambdaExecute*” de ser necesario se puede escoger también la política de Full acceso.

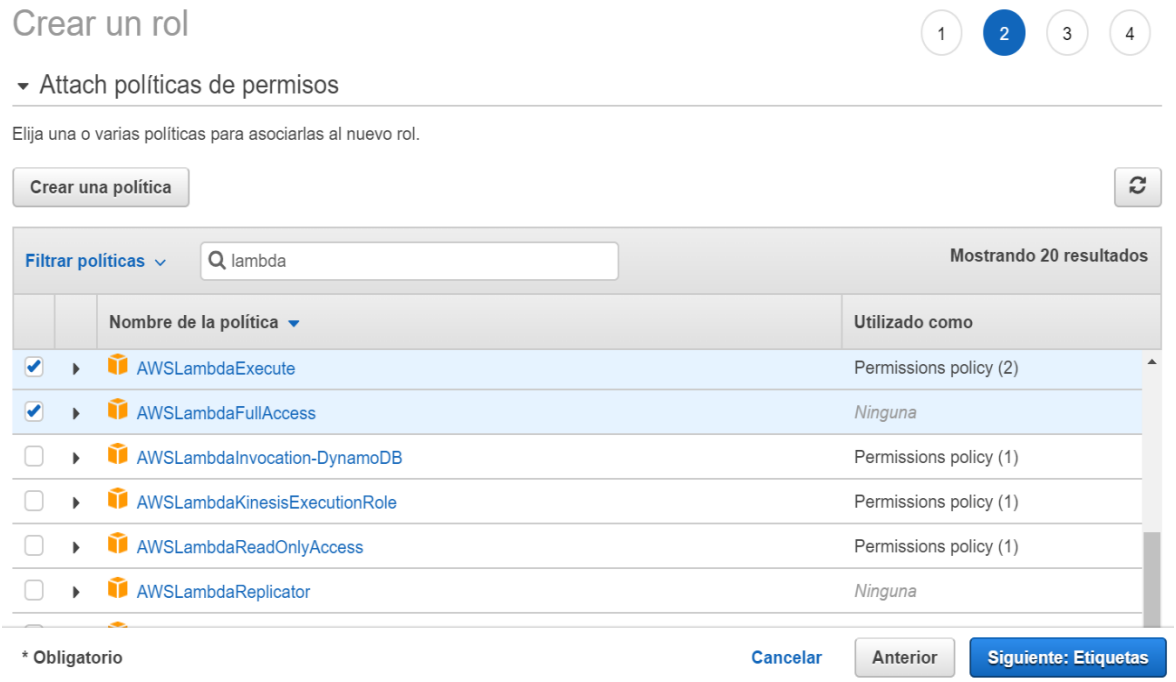


Figura 16. Creación de Rol, selección de políticas.

El tercer paso para la creación de un rol es opcional. Se puede añadir etiquetas al rol que pueden incluir información del usuario como por ejemplo información descriptiva o dirección de correo electrónico. Estas etiquetas sirven para organizar de mejor manera el acceso al rol.

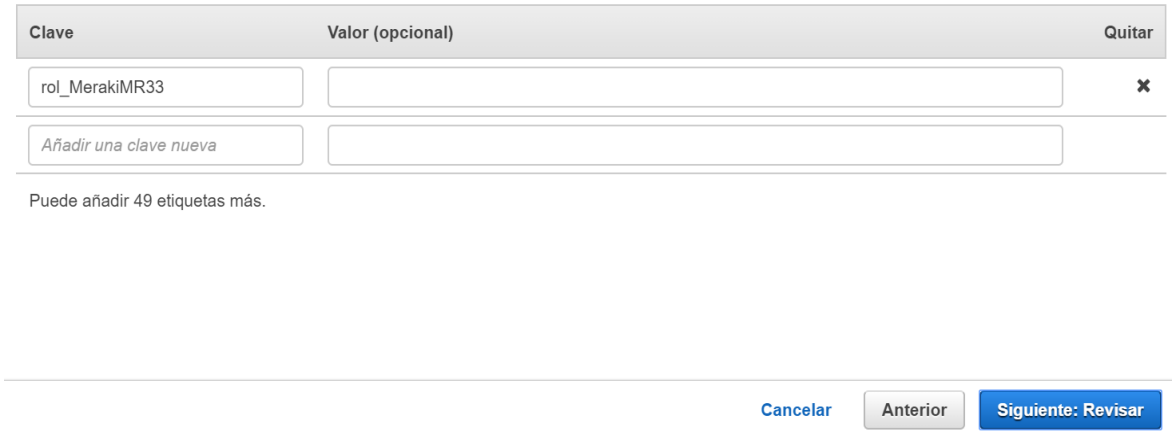


Figura 17. Creación de Rol, añadir etiquetas.

Como último paso, se debe ingresar el nombre que tendrá el rol, una descripción de lo que puede hacer y también se puede revisar si las políticas seleccionas están correctas.

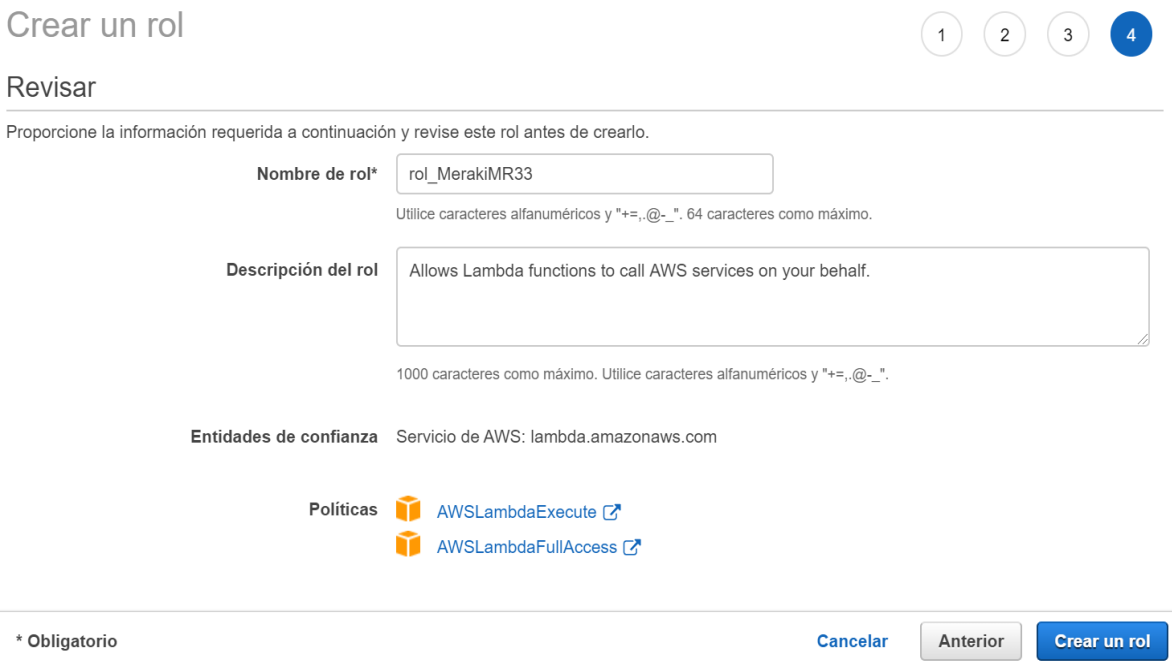


Figura 18. Creación de Rol.

3.4.3. Creación de Función en Lambda

Para poder crear una nueva función de Lambda se debe ingresar a la pestaña de servicios y seleccionar “Lambda”. Posteriormente dar clic izquierdo en la opción de “Crear una función” como se muestra en las figuras 19 y 20.

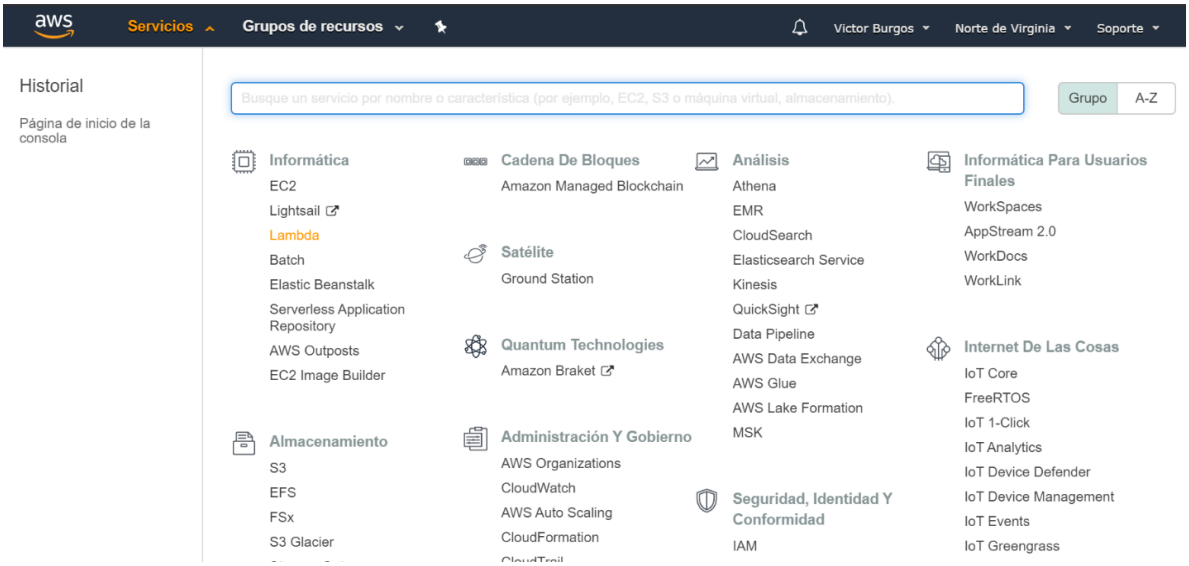


Figura 19. Selección de servicios.

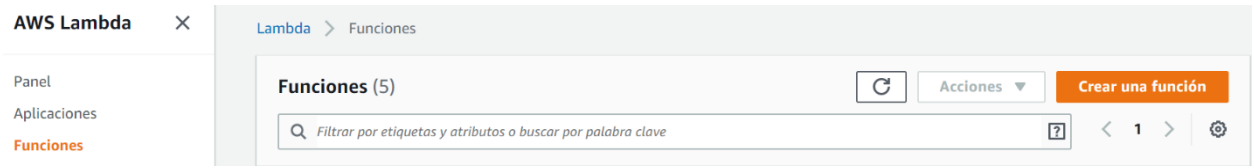


Figura 20. Creación de Función Lambda.

Para la creación de la función, primero se debe escoger crear desde cero, llenar la información básica como asignar un nombre para la función y seleccionar el tipo de lenguaje de programación que tendrá el código que se vaya a usar.

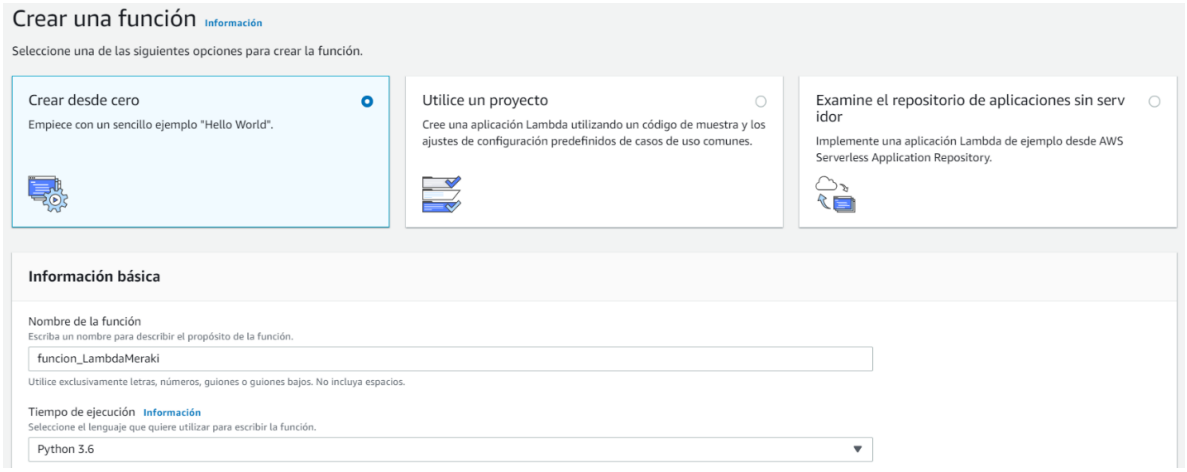


Figura 21. Datos para la creación de Función Lambda.

Para completar la creación de la función lambda es necesario asignar permisos. Para ello, hay que marcar la opción de uso de rol existente y seleccionar el rol creado anteriormente.

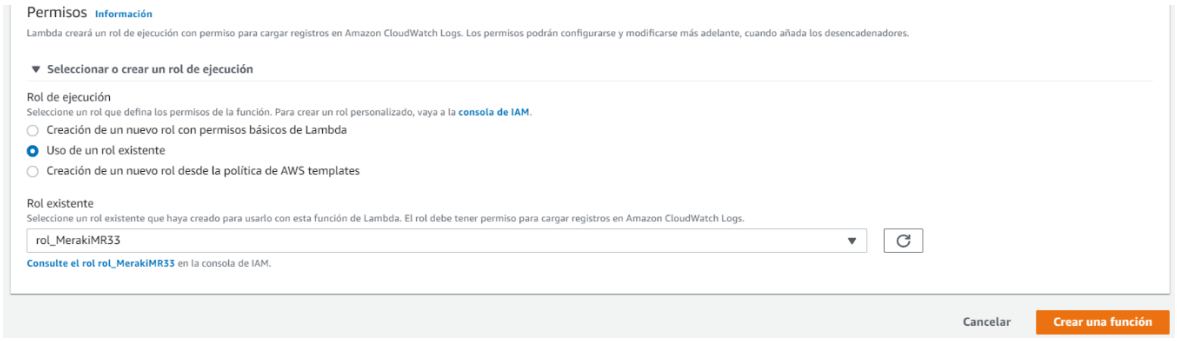


Figura 22. Creación de Función Lambda, selección de rol.

Creada la función se despliega una pantalla similar a la figura 23. Donde se puede revisar las configuraciones, los permisos y realizar una monitorización.

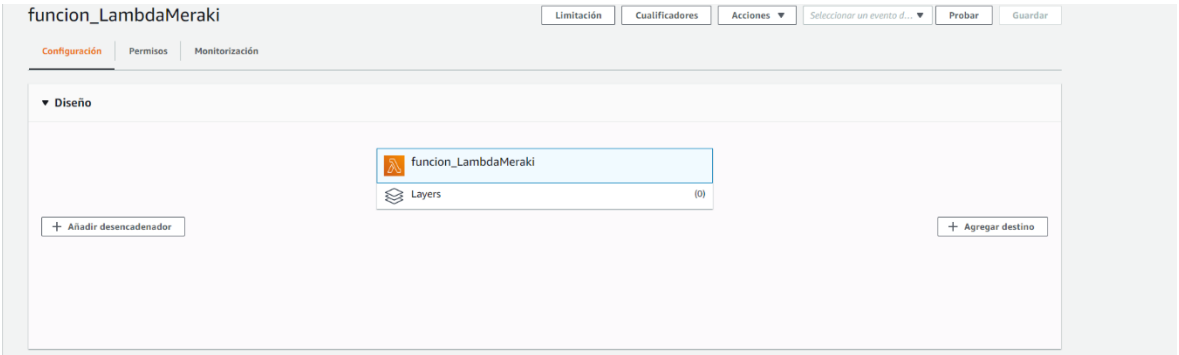


Figura 23. Función Lambda.

En la opción de permisos se debe verificar que se cuenta con el rol de ejecución y con siguientes servicios:

- Amazon S3.
- Amazon CloudWatch Logs.

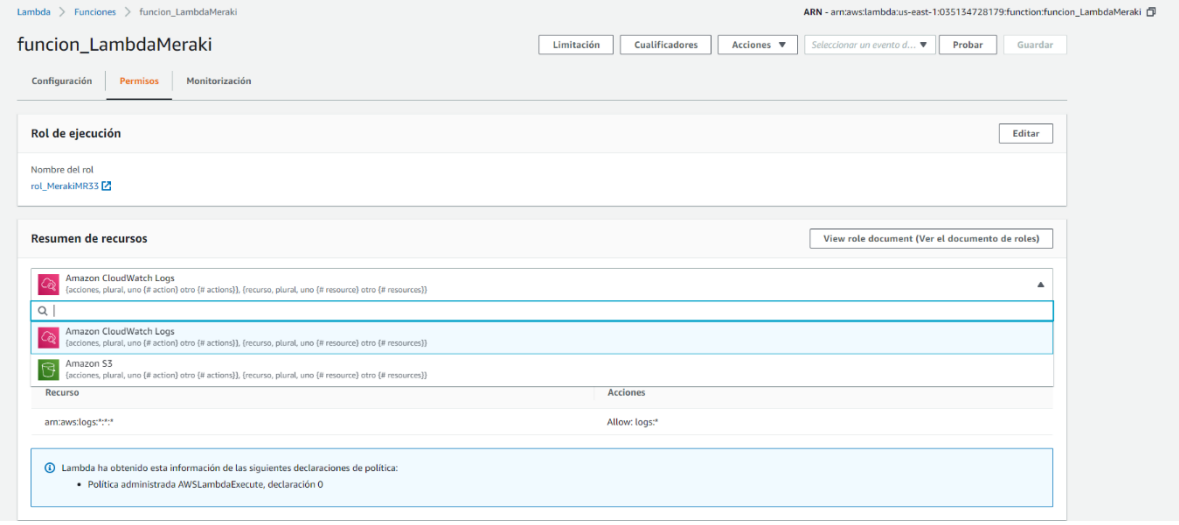


Figura 24. Revisión de permisos.

Para cargar el código de Python hay que dirigirse al código de la función y seleccionar cargar un archivo de tipo zip, donde se debe escoger el tipo de lenguaje para este proyecto de titulación se hizo en Python 3.6 y por último escribir el nombre del controlador.

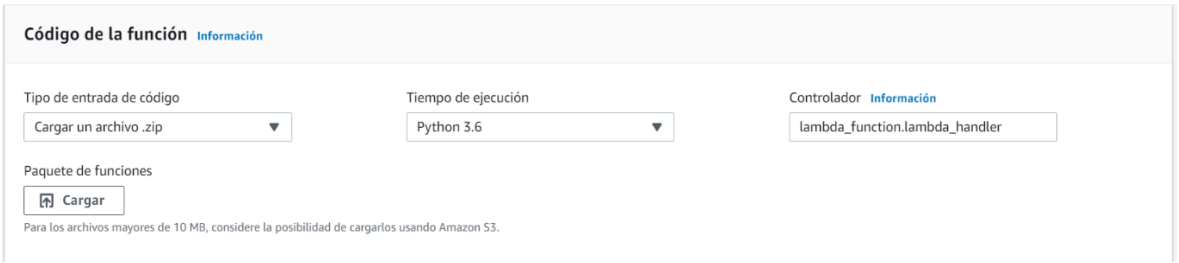


Figura 25. Cargar el código.

Una vez cargado el código, se debe abrir el archivo “meraki_info.py” y agregar los parámetros del API Key, ID de la organización, ID de la red, numero serial del dispositivo, entre otros datos. Estos datos se los puede obtener del *dashboard* del Meraki como de Postman.

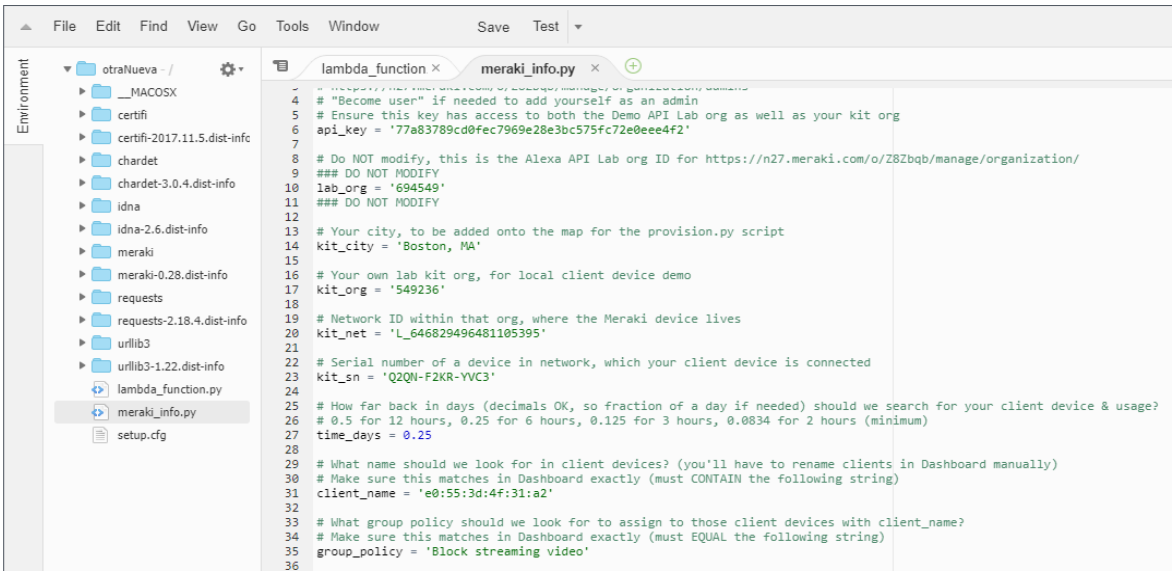


Figura 26. Configuración de variables.

Para poder realizar la integración de la plataforma AWS con la habilidad de Amazon Alexa hay que añadir un desencadenador como se enseña en la figura 27. Además, se recomienda deshabilitar el ID de habilidad ya que suele dar errores.

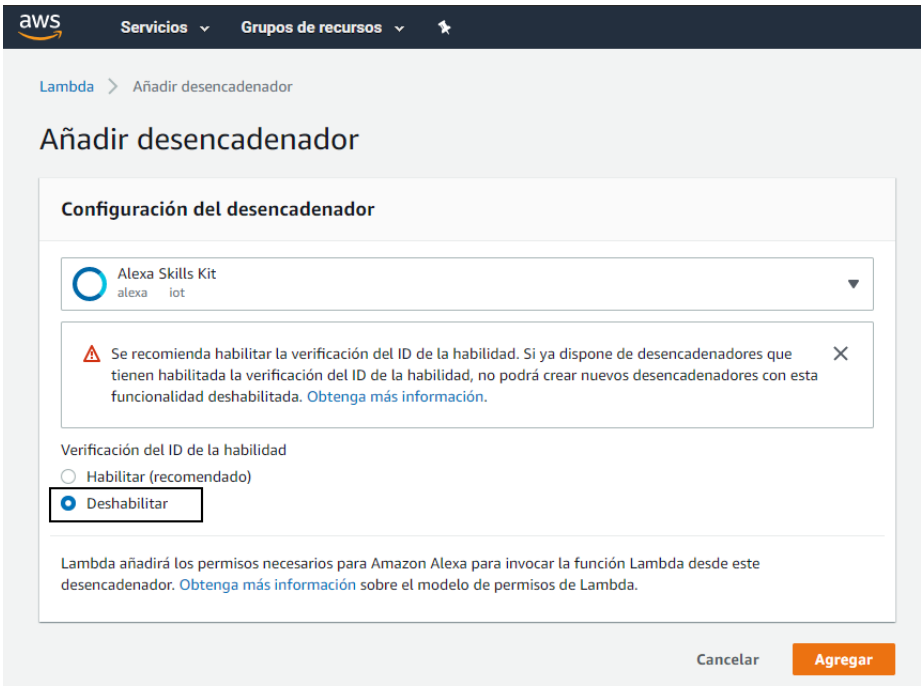


Figura 27. Agregar Desencadenador.

Cuando se tenga cargado el código, verificados los permisos y añadido el desencadenador. Se debe copiar el ARN que se encuentra en la esquina superior derecha de la página. Ya que este se usa en la plataforma de Amazon Alexa para poder realizar la conexión con AWS.

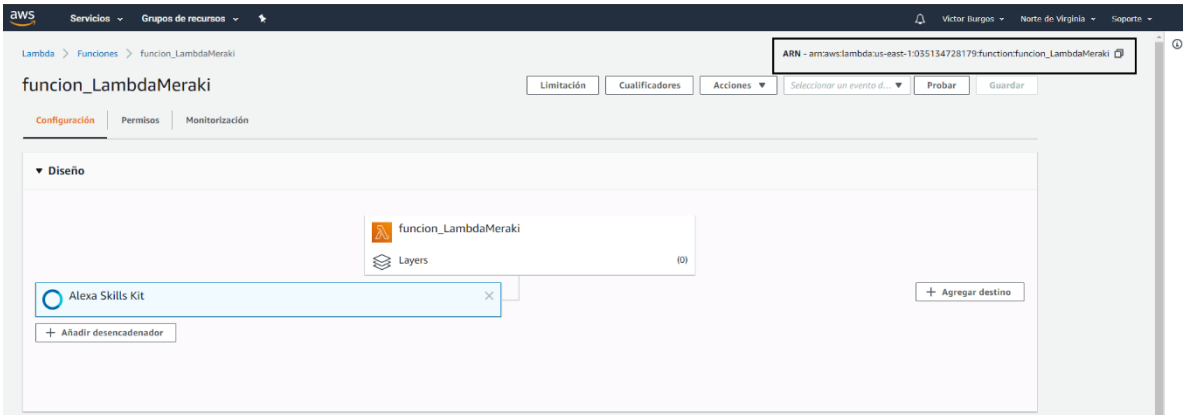


Figura 28. Nombre de Recursos de Amazon (ARN).

3.5. Configuración e Integración de Alexa con AWS

Para crear una habilidad en Amazon Alexa se debe ingresar al siguiente enlace https://developer.amazon.com/en-US/alexa/alexa-skills-kit y dar clic sobre “Consola”.

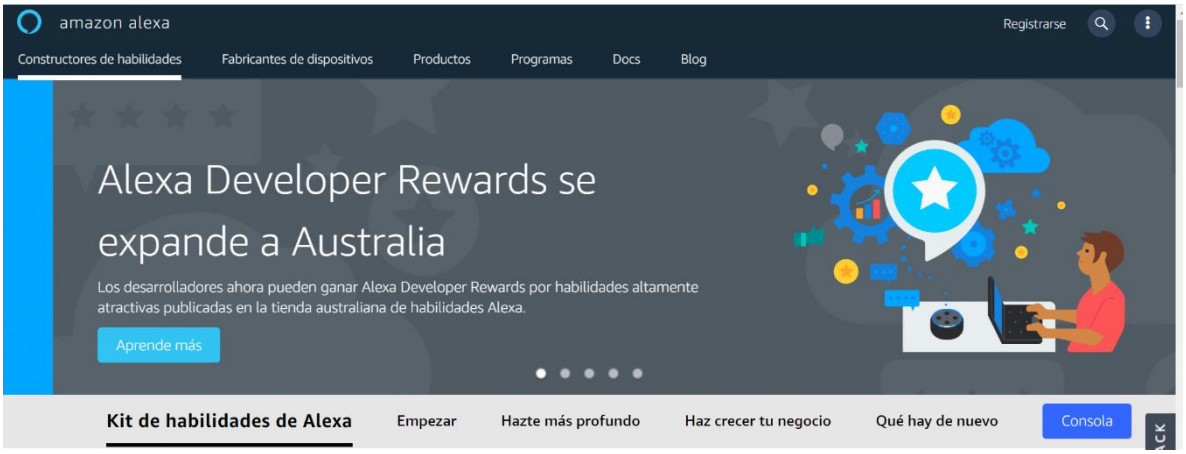


Figura 29. Plataforma de Amazon Alexa.

La cuenta de Amazon Alexa es diferente a la cuenta de Amazon Web Services. Por lo tanto, si no se tiene una cuenta debe ser creada.

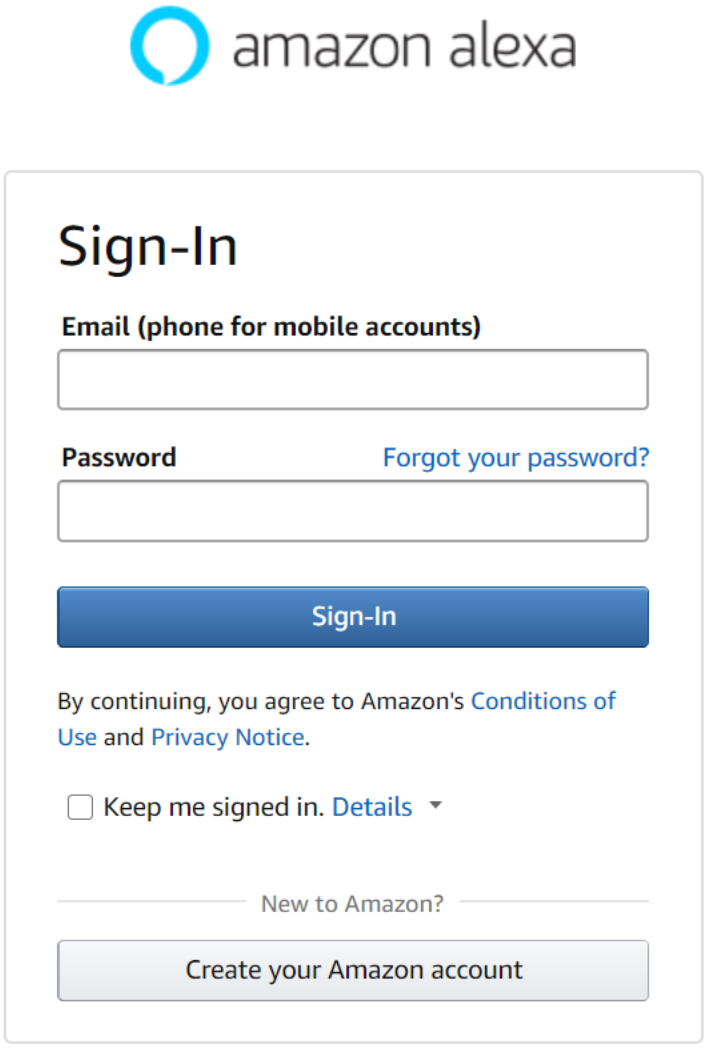


Figura 30. Ingreso a la plataforma de Alexa.

Una vez registrado, se puede crear habilidades dando clic izquierdo sobre "*Create Skill*" tal como se muestra en la figura 31.

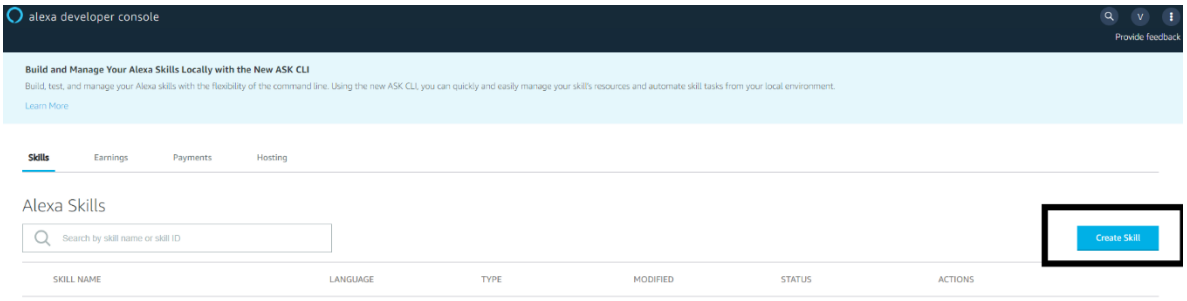


Figura 31. Creación de habilidad.

Para crear una habilidad de se debe asignar un nombre y además el idioma con el que se quiere interactuar por medio de comandos de voz.

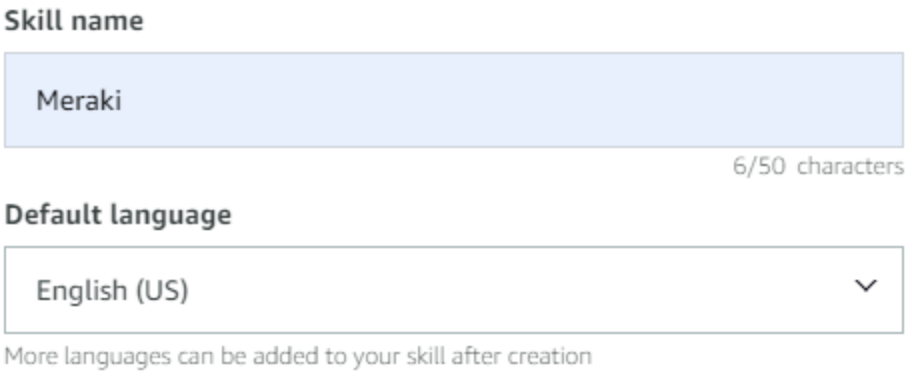


Figura 32. Ingreso de nombre e idioma.

Amazon Alexa brinda la posibilidad de escoger entre varios modelos y métodos de *backend* para iniciar con la creación de una habilidad. Para este proyecto de titulación se escogió la opción de personalizar uno mismo.

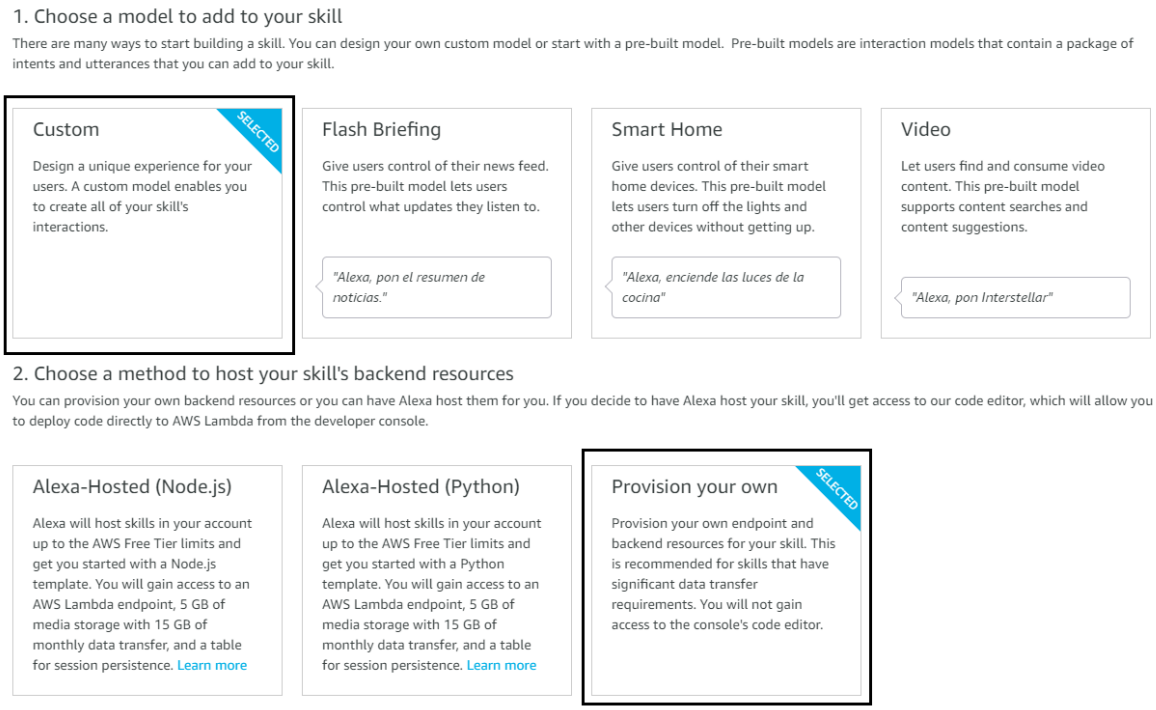


Figura 33. Selección de modelos.

Se debe agregar un nombre para la invocación, ya que con este nombre el usuario activa la habilidad. El nombre debe ser simple y mínimo de dos palabras por ejemplo "*hello cisco*". Por último, se debe guardar dando clic en el botón de la parte superior.

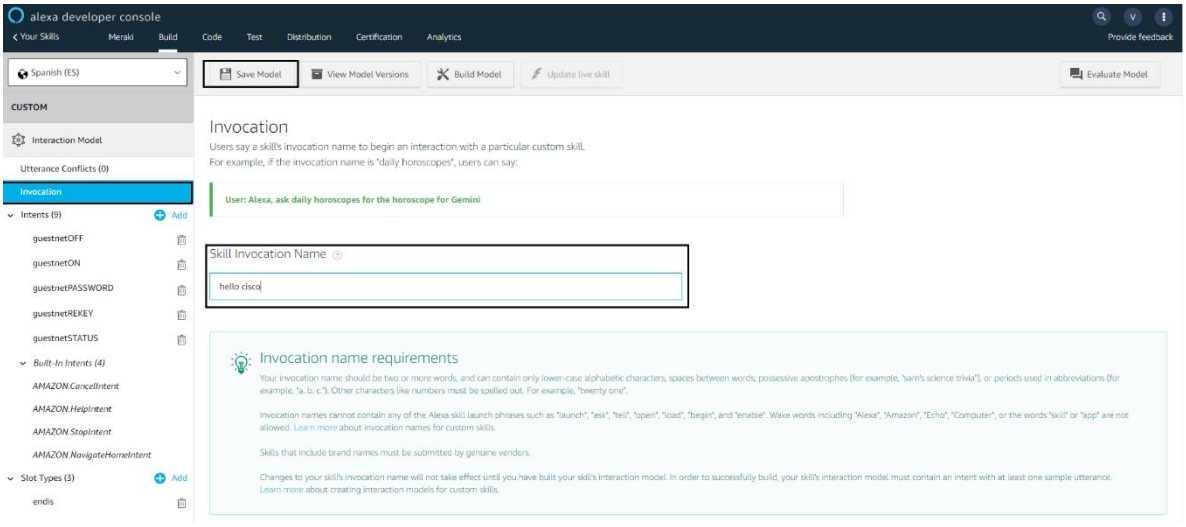


Figura 34. Nombre de invocación.

Para el *endpoint* se debe pegar el ARN que se copió de la función lambda de la plataforma de AWS en el aparatado de región por defecto. Después de cualquier cambio se debe dar clic en guardar como se muestra en la figura 35.

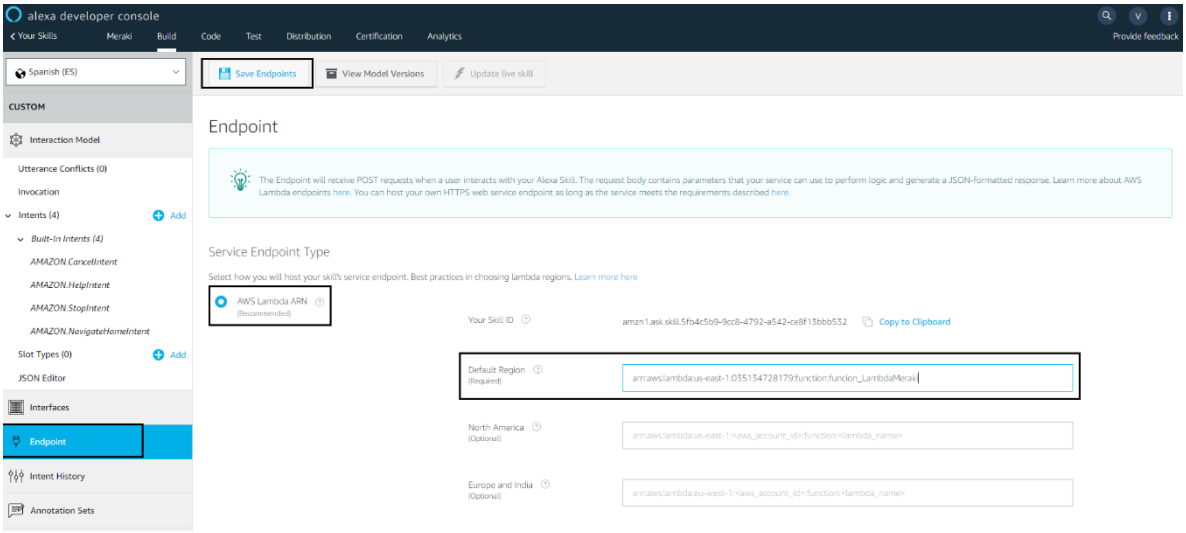


Figura 35. Ingreso de *Endpoint ARN.*

En la opción de “JSON Editor” se importa el esquema de intención Meraki. Posteriormente se debe dar clic en guardar modelo y construir modelo, de esta manera se genera la habilidad.

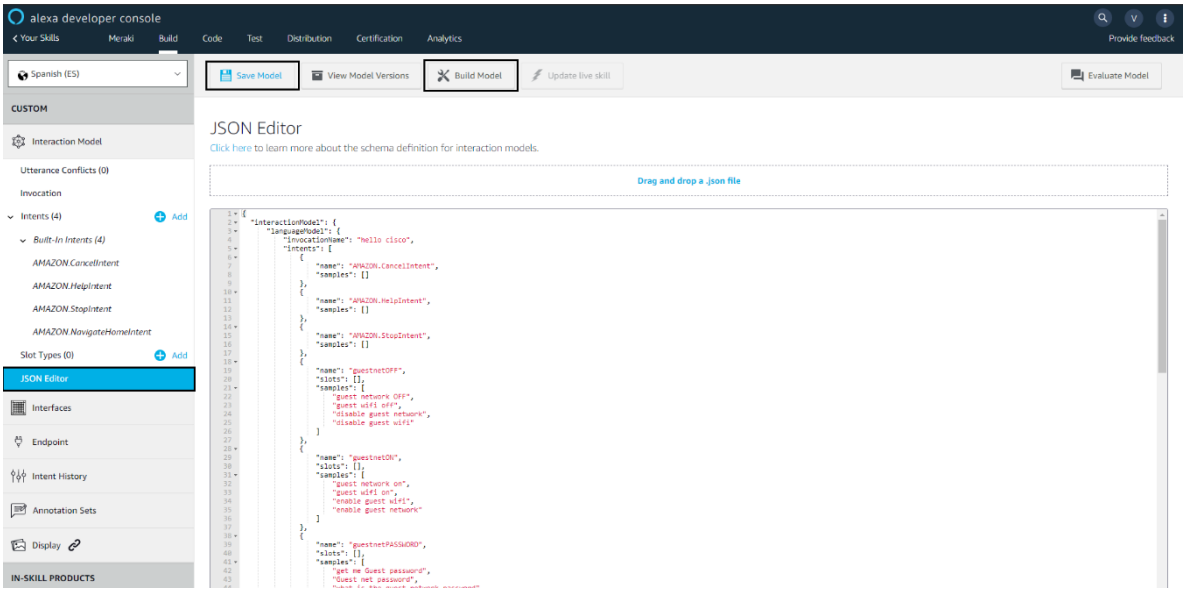


Figura 36. Ingreso de JSON Editor.

Finalmente, después de añadir el nombre de invocación, *endpoint* y *json* se puede ver una pantalla similar a la figura 37. Donde se muestra que todos los pasos están correctos y se puede realizar las pruebas de la habilidad creada.

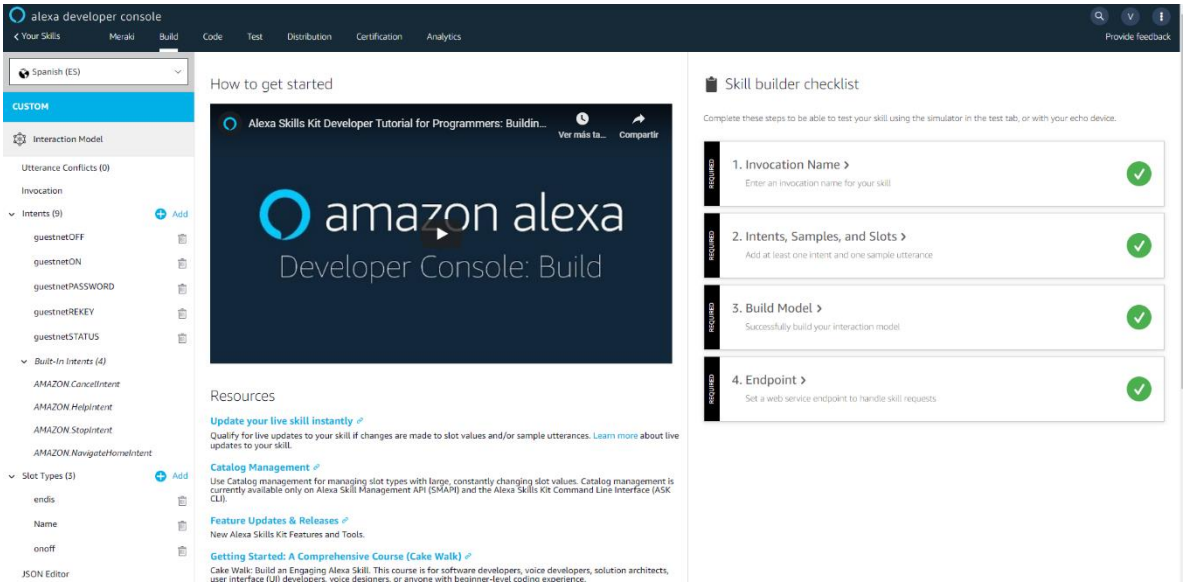


Figura 37. Comprobación de datos ingresados.

3.6. Creación de código Python

Se adjunta el código que es utilizado para interacción de las plataformas previamente mencionadas. Fue escrito en el lenguaje de programación Python y cuenta con algunas características en particular. Este código contiene diferentes métodos para la obtención de datos en el dispositivo.

El siguiente método consiste en mostrar el de mensaje de bienvenida y el de sesión culminada.

```
def get_welcome_response():
    """ If we wanted to initialize the session to have some attributes we could
    add those here
    """

    session_attributes = {}
    card_title = "Welcome"
    speech_output = "Welcome to the Alexa Meraki skill. " \
                    "You can ask me to get network status, check license expiration, " \
                    "query device status, get client devices."
    # If the user either does not reply to the welcome message or says something
    # that is not understood, they will be prompted again with this text.
    reprompt_text = "Please tell me a command, " \
                    "like asking why the Internet is slow."
    should_end_session = False
    return build_response(session_attributes, build_speechlet_response(
        card_title, speech_output, reprompt_text, should_end_session))


def handle_session_end_request():
    card_title = "Session Ended"
    speech_output = "Thank you for trying the Alexa Skills Kit sample. " \
                    "Have a nice day! "
    # Setting this to true ends the session and exits the skill.
    should_end_session = True
    return build_response({}, build_speechlet_response(
        card_title, speech_output, None, should_end_session))
```

Figura 38. Mensaje de bienvenida y sesión culminada.

El siguiente método es para realizar una pregunta y que se pueda identificar el dispositivo con el cual se encuentra trabajando, en este caso Meraki.

```
def who_are_you(intent, session):
    session_attributes = {}
    reprompt_text = None

    speech_output = "I am the omniscient Cisco Meraki cloud."
    should_end_session = True


    # Setting reprompt_text to None signifies that we do not want to reprompt
    # the user. If the user does not respond or says something that is not
    # understood, the session will end.
    return build_response(session_attributes, build_speechlet_response(
        intent['name'], speech_output, reprompt_text, should_end_session))
```

Figura 39. Método quién eres.

Aquí se realiza la petición del estado de licencia, aquí se muestra si cuenta actualmente con licencia o ha expirado.

```
def get_license_status(intent, session):
    session_attributes = {}
    reprompt_text = None

    licensing = meraki.getlicensestate(API_KEY, KIT_ORG)
    status = licensing['status']
    date = licensing['expirationDate'].strip(' UTC')
    days = (datetime.strptime(date, '%b %d, %Y') - datetime.today()).days

    speech_output = 'Your license is {0} and expires on {1}, which is {2} days from today.'.format(status, date, days)
    should_end_session = True


    # Setting reprompt_text to None signifies that we do not want to reprompt
    # the user. If the user does not respond or says something that is not
    # understood, the session will end.
    return build_response(session_attributes, build_speechlet_response(
        intent['name'], speech_output, reprompt_text, should_end_session))
```

Figura 40. Método revisión de licencia.

El siguiente método comprueba el estado de la red, con la finalidad de mostrar si se encuentra activo.

```
def get_network_status(intent, session):
    session_attributes = {}
    reprompt_text = None

    network = meraki.getnetworkdetail(API_KEY, KIT_NET)
    devices = meraki.getnetworkdevices(API_KEY, KIT_NET)
    device_uplinks = [meraki.getdeviceuplink(API_KEY, KIT_NET, device['serial']) for device in devices]
    active = 0
    for uplinks in device_uplinks:
        for uplink in uplinks:
            if uplink['status'] == 'Active':
                active += 1
                break
    if len(devices) == 1:
        speech_output = 'The {0} network contains just a single device, and that device is currently {1} active and online.'.format(network['name'],
        ('not' if active == 0 else ''))
    else:
        speech_output = 'The {0} network contains a total of {1} devices, and {2} of them {3} currently active and online.'.format(network['name'], len(devices), active,
    ('is' if active == 1 else 'are'))
    should_end_session = True

    # Setting reprompt_text to None signifies that we do not want to reprompt
    # the user. If the user does not respond or says something that is not
    # understood, the session will end.
    return build_response(session_attributes, build_speechlet_response(
        intent['name'], speech_output, reprompt_text, should_end_session))
```

Figura 41. Método estado red.

Este método tiene como propósito revisar el estado del Meraki, comprobando si está encendido o fuera de línea.

```
def get_device_status(intent, session):
    session_attributes = {}
    reprompt_text = None

    device = meraki.getdevicedetail(API_KEY, KIT_NET, KIT_SN)
    uplinks = meraki.getdeviceuplink(API_KEY, KIT_NET, KIT_SN)
    state = 'offline'
    for uplink in uplinks:
        if uplink['status'] == 'Active':
            state = 'online'
            break
    speech_output = 'The {0} device is a {1} located at {2}, and is currently {3}.'.format((device['name'] if device['name'] != '' else 'Meraki'), device['model'], (device['address'] if
    should_end_session = True

    # Setting reprompt_text to None signifies that we do not want to reprompt
    # the user. If the user does not respond or says something that is not
    # understood, the session will end.
    return build_response(session_attributes, build_speechlet_response(
        intent['name'], speech_output, reprompt_text, should_end_session))
```

Figura 42. Método estado dispositivo.

El siguiente método consiste en indicar los dispositivos que se encuentran conectados al Meraki.

```
def get_client_devices(intent, session):
    session_attributes = {}
    reprompt_text = None

    clients = meraki.getclients(API_KEY, KIT_SN, timestamp=3600)
    usage = 0
    for client in clients:
        usage += client['usage']['sent'] + client['usage']['recv']
    if usage < 1024:
        speech_output = 'In the last hour there are a total of {0} client devices connected, with total network usage of {1} kilobytes.'.format(len(clients), round(usage))
    elif usage < 1024 * 1024:
        speech_output = 'In the last hour there are a total of {0} client devices connected, with total network usage of {1} megabytes.'.format(len(clients), round(usage/1024, 1))
    else:
        speech_output = 'In the last hour there are a total of {0} client devices connected, with total network usage of {1} gigabytes.'.format(len(clients), round(usage/1024/1024, 2))
    should_end_session = True

    # Setting reprompt_text to None signifies that we do not want to reprompt
    # the user. If the user does not respond or says something that is not
    # understood, the session will end.
    return build_response(session_attributes, build_speechlet_response(
        intent['name'], speech_output, reprompt_text, should_end_session))
```

Figura 43. Método dispositivos clientes.

Este método es para borrar todas las redes registradas en el dispositivo, sirve para ingresar un nuevo registro de las redes.

```
def delete_all_networks(intent, session):
    session_attributes = {}
    reprompt_text = None

    speech_output = ''
    ### Do NOT NOT NOT change the following org ID or else you will accidentally delete something but the lab!
    networks = meraki.getnetworklist(API_KEY, LAB_ORG)

    speech_output = ''
    for network in networks:
        meraki.delnetwork(API_KEY, network['id'])
        speech_output += 'Goodbye ' + network['name'] + '. '
    speech_output += 'Have a nice day!'
    should_end_session = True
    return build_response(session_attributes, build_speechlet_response(
        intent['name'], speech_output, reprompt_text, should_end_session))
```

Figura 44. Método borrar redes.

4. Capitulo IV. Migración del sistema para la administración del UCS al lenguaje de programación Python 3.7

A continuación, se adjunta el código del sistema de administración mediante Amazon Alexa para un UCS, este se encuentra actualmente instalado en el data center de la Universidad de las Américas.

El sistema implementado cuenta con la versión de Python 2.7 y en la actualidad la versiones 2.X ya no tienen soporte en caso de presentar fallas, por tal motivo se tiene que migrar a las nuevas versiones 3.X.

En las nuevas versiones de Python 3.X se realizan cambios en el código, entre esas tenemos las siguientes.

Función Print

Este es la diferencia más conocida en las versiones de Python 2.X a 3.X. En la versión 2.X para imprimir algún objeto no era necesario colocar paréntesis para imprimir el objeto, ahora para la versión 3.X es necesario colocar el objeto en paréntesis para realizar la impresión.

División Números Enteros

Al dividir en la versión 2.X números enteros este daba como resultado un numero entero, por ende, si realizaba una división de ½ daba como resultado 0, con la versión 3.X aquí al realizar la misma operación da como resultado ya con los números decimales.

Funciones Input

Esta función captura los datos que son ingresados para realizar operaciones, en la versión 2.X hay 2 formas para el ingreso debido a que con el *input* simple únicamente tomar valores de números, en el input complejo *raw_input* toma datos con caracteres o *strings*. En la versión 3.X únicamente se dejó el input para que esta coja datos numéricos y cadenas de *strings*.

Comparación de datos

Se podía realizar la comparación entre datos en la versión 2.X en este caso un ejemplo podría ser “2 < '3'” da como valor *true*, pero en la versión 3.X esto no puede ser realizado, bota un error.

Iterar Datos

En las versiones 2.X realizar iteraciones es posible por medio de 2 funciones *iteritems*() o ítems, en la versión 3.X para realizar las acciones repetitivas únicamente se pude hacer con *ítems*(), porque al querer usar *iteritems*() da un error de atributo.

Funciones Range Xrange

Estas funciones se realizan para la creación de listas de enteros o para la creación con los *bucles for* esto se lo puede hacer con las versiones 2.X, pero para la versión 3.X únicamente se puede usar la función *range*().

Módulo future

Esto permite colocar comandos que no son posibles en las versiones que se tiene en Python, un ejemplo es adaptar la funcionalidad de *print* de la versión 3.X a la versión 2.X.

Se adjunta 2 códigos con el detalle de lo que hacen los métodos, el uno con las habilidades de Alexa y el otro con las operaciones para el UCS. ("What's New In Python 3.7 — Python 3.7.7 documentation", 2020).

DevNet Skill

La funcionalidad que tiene la figura 45 es llamar a la función e inicie la sesión, además que en la función *print* se cambia el código de la versión 2.7 a la versión 3.7 de Python añadiendo los paréntesis.

```
def on_session_started(session_started_request, session):
    """ Called when the session starts """

    print(("on_session_started requestId=" + session_started_request['requestId']
          + ", sessionId=" + session['sessionId']))
```

Figura 45. Obtención de perfil.

Dentro de los cambios realizados para la migración en los métodos *on_launch*, *on_intent*, *on_session_ended* y lambda *handler* se lo realizó en las funciones *print*, aquí han sido agregados un paréntesis al inicio y al final de la función como se lo mencionó en los detalles del código en las versiones 3.X.

```
def on_launch(launch_request, session):
    """ Called when the user launches the skill without specifying what they
    want
    """

    print(("on_launch requestId=" + launch_request['requestId'] +
          ", sessionId=" + session['sessionId']))
    # Dispatch to your skill's launch
    return get_welcome_response()
```

Figura 46. Inicio de Habilidades.

En la funcionalidad del método de la figura 47 es llamar a una habilidad en específico que es solicitada por el usuario.

```
def on_intent(intent_request, session):
    """ Called when the user specifies an intent for this skill """

    print(("on_intent requestId=" + intent_request['requestId'] +
          ", sessionId=" + session['sessionId']))

    intent = intent_request['intent']
    intent_name = intent_request['intent']['name']
```

Figura 47. Habilidad.

Este método es encargado de solicitar al usuario cuando culmine la sesión.

```
def on_session_ended(session_ended_request, session):
    """ Called when the user ends the session.

    Is not called when the skill returns should_end_session=true
    """
    print(("on_session_ended requestId=" + session_ended_request['requestId'] +
          ", sessionId=" + session['sessionId']))
    # add cleanup logic here
```

Figura 48. Sesión culminada.

De igual manera que en la figura 46 se realiza el cambio en la función *print*, agregando los paréntesis a los costados respectivos.

```
def lambda_handler(event, context):
    """ Route the incoming request based on type (LaunchRequest, IntentRequest,
    etc.) The JSON body of the request is provided in the event parameter.
    """
    print(("event.session.application.applicationId=" +
          event['session']['application']['applicationId']))
```

Figura 49. Código migrado.

En la figura 50 se observa 2 métodos, el uno es encargado de la creación de *Vlans* en el UCS por medio de Alexa, y el otro método es para borrar las *Vlans*.

```
# Add a UCS VLAN
def add_vlan(intent, session):
    session_attributes = {}
    reprompt_text = None

    speech_output = ucsm_operations.add_ucs_vlan(intent['slots']['vlan_id']['value'])
    should_end_session = True

    return build_response(session_attributes, build_speechlet_response(
        intent['name'], speech_output, reprompt_text, should_end_session))

# Remove a UCS VLAN
def remove_vlan(intent, session):
    session_attributes = {}
    reprompt_text = None

    speech_output = ucsm_operations.remove_ucs_vlan(intent['slots']['vlan_id']['value'])
    should_end_session = True

    return build_response(session_attributes, build_speechlet_response(
        intent['name'], speech_output, reprompt_text, should_end_session))
```

Figura 50. Crear, Remover Vlans.

El siguiente método tiene como propósito obtener el perfil para la asociación del UCS con Alexa.

```
# Create and Associate a Service Profile to an available server
def set_server(intent, session):
    session_attributes = {}
    reprompt_text = None

    speech_output = result = ucsm_operations.set_ucs_server()
    should_end_session = True

    return build_response(session_attributes, build_speechlet_response(
        intent['name'], speech_output, reprompt_text, should_end_session))
```

Figura 51. Obtención de perfil.

En la figura 52 se puede observar el método de validación para el ingreso a la interfaz de la UCS.

```
def ucsm_login():

    global handle, status

    username = [redacted]
    password = [redacted]

    try:
        handle = UcsHandle(ucsmhost,username,password)
        if handle.login() == True:
            status['login'] = "success"
            return

    except urllib2.URLError as err:
        status['login'] = "URLError"
        return

    except UcsException as err:
        status['login'] = "UcsException"
        return
```

Figura 52. Validación Ingreso UCS.

Para el resto del código se mantiene igual debido a que en la migración a la nueva versión no se realiza más cambios.

5. Capitulo V. Pruebas y Resultados

Para la obtención de los datos del dispositivo Meraki se utilizó la plataforma de Postman, este *software* recolecta la información del Meraki por medio de métodos. Los datos que se obtienen se utilizan en el código de Python para vincular el Meraki con el resto de las plataformas.

Para poder vincular el Meraki con Postman se necesita ingresar el API Key y la URL que identifica al dispositivo. Por medio de métodos se solicita las organizaciones asociadas del Meraki como se muestra en la figura 53.

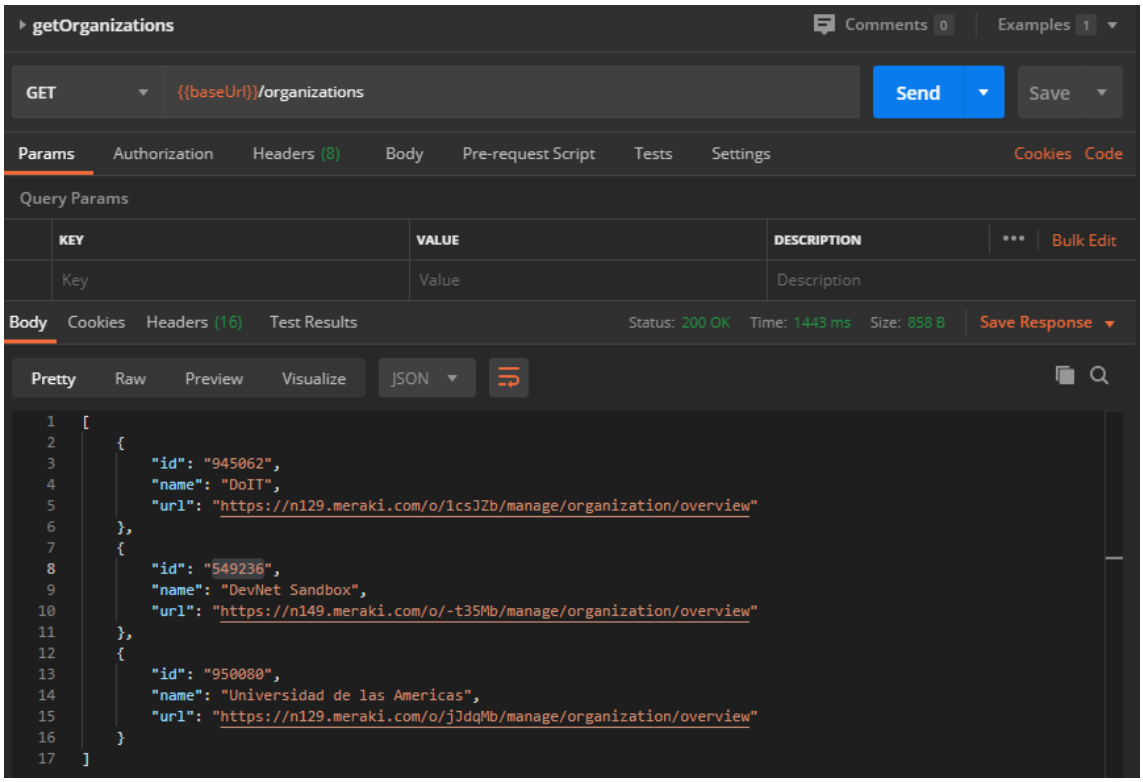


Figura 53. Método ID Organizaciones.

Por medio del siguiente método se obtienen las redes que tiene la organización. En base al ID de la organización que se obtuvo con el método de la figura 53, se puede identificar cual es el ID de la red del Meraki que se esté utilizando.

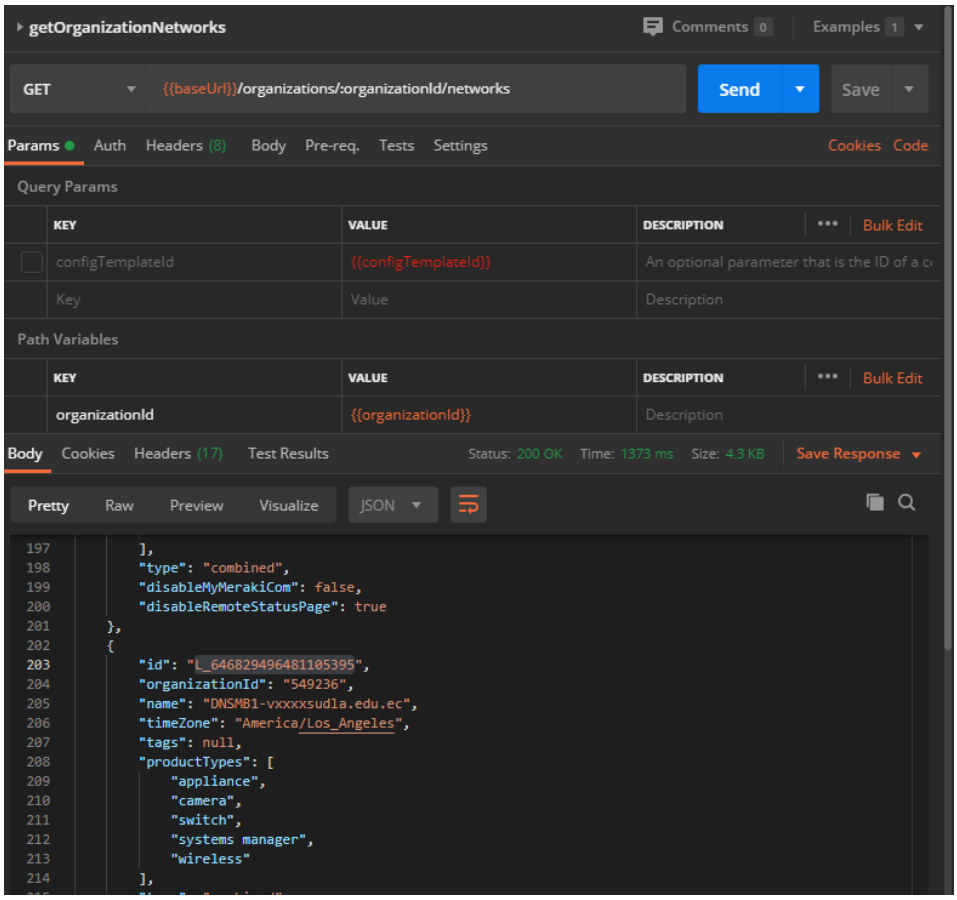


Figura 54. Obtención ID de Red.

En la figura 55 se obtiene los clientes que se encuentran conectados al dispositivo, así como el ID, MAC, IP y otros datos relacionados a los clientes.

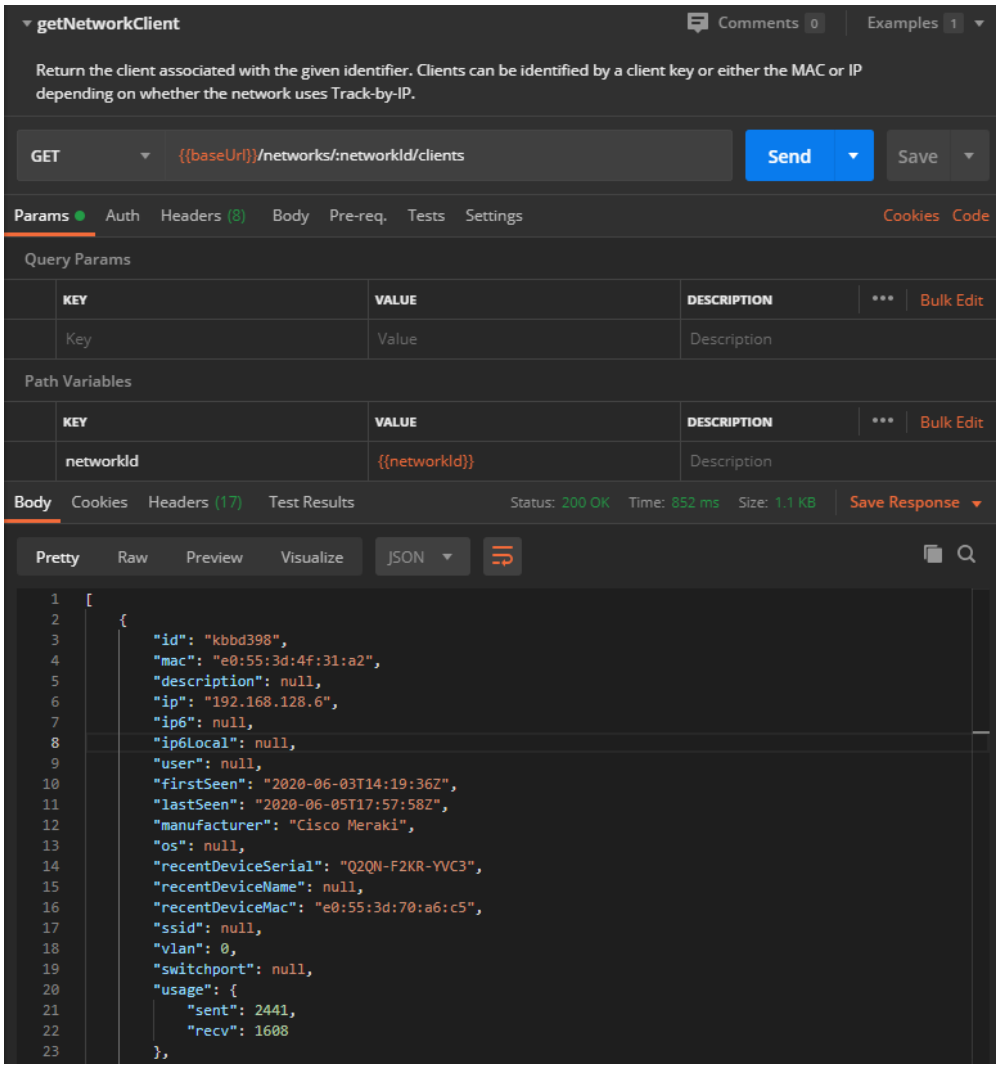


Figura 55. Método ID Clientes.

En el siguiente método se obtiene los dispositivos que se encuentran conectados a la red, también se pueden ver otros datos como la ubicación, seriales referentes a los dispositivos conectados.

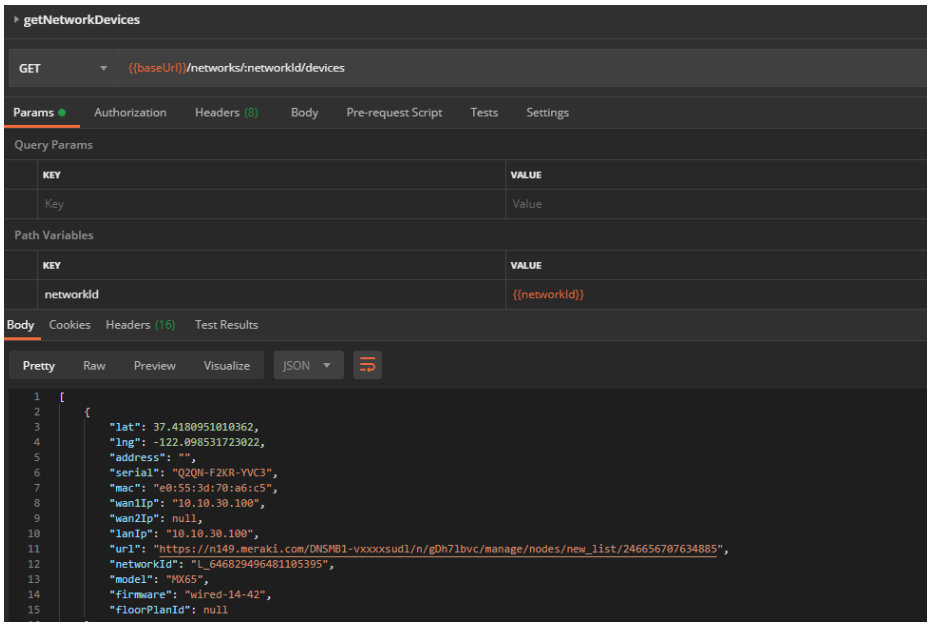


Figura 56. Obtención de dispositivos en la red.

En la figura 57 se muestra el método de Network ID, con este se obtiene todos los datos de la red solicitada.

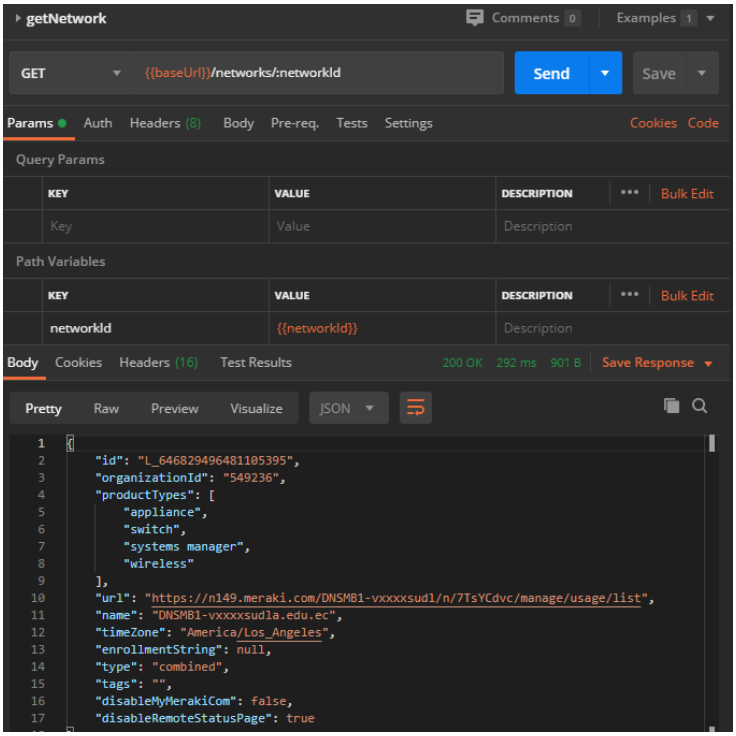


Figura 57. Datos de red.

Las siguientes imágenes fueron realizadas en la plataforma de Amazon Alexa, esta permite emular un Echo Dot físico y realizar las pruebas necesarias, verificando con éxito la implementación del proyecto.

En la figura 58 se realiza el ingreso al dispositivo Meraki por medio del llamado de invocación "*hello cisco*", se puede comprobar la funcionalidad y el ingreso al dispositivo cuando Alexa contesta con el mensaje de bienvenida colocado en el código de Python.

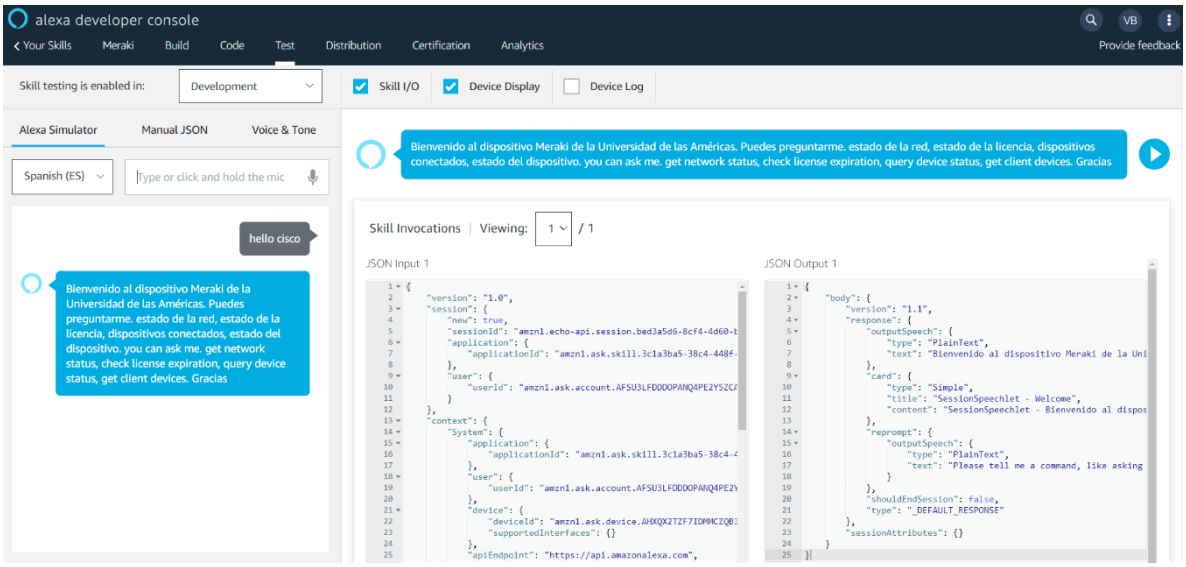


Figura 58. Ingreso al Meraki.

Para poder realizar cualquier petición al Meraki primero se debe hacer el llamado de invocación y después la solicitud que se quiere realizar.

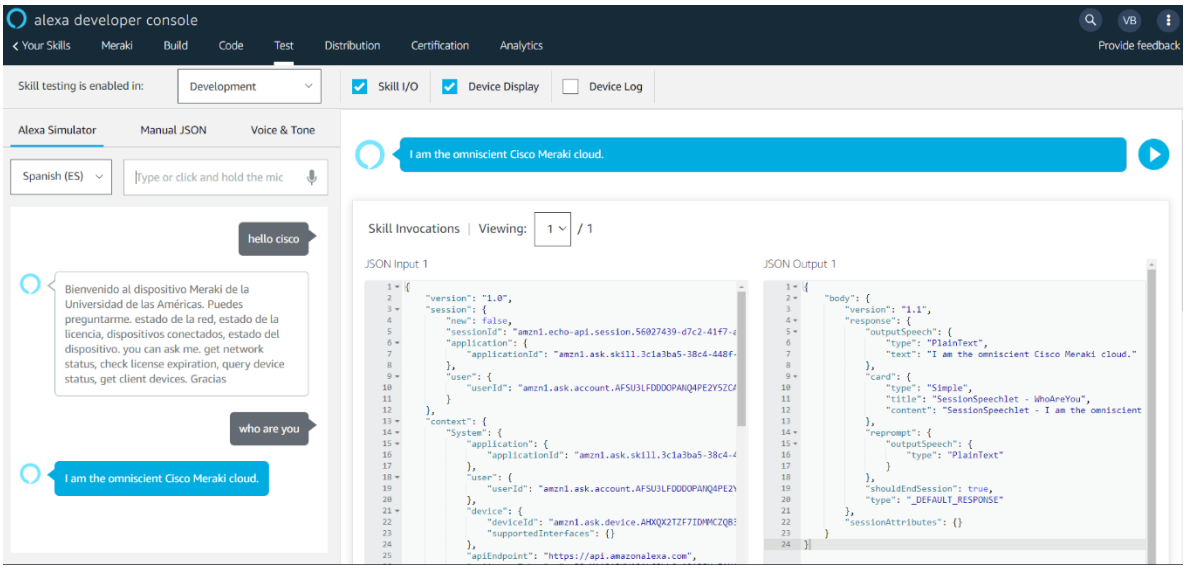


Figura 59. Prueba de funcionalidad.

Como se muestra en la figura 60, se solicita el estado de la red, devolviendo como resultado el nombre de la red, el total de dispositivos y los dispositivos conectados.

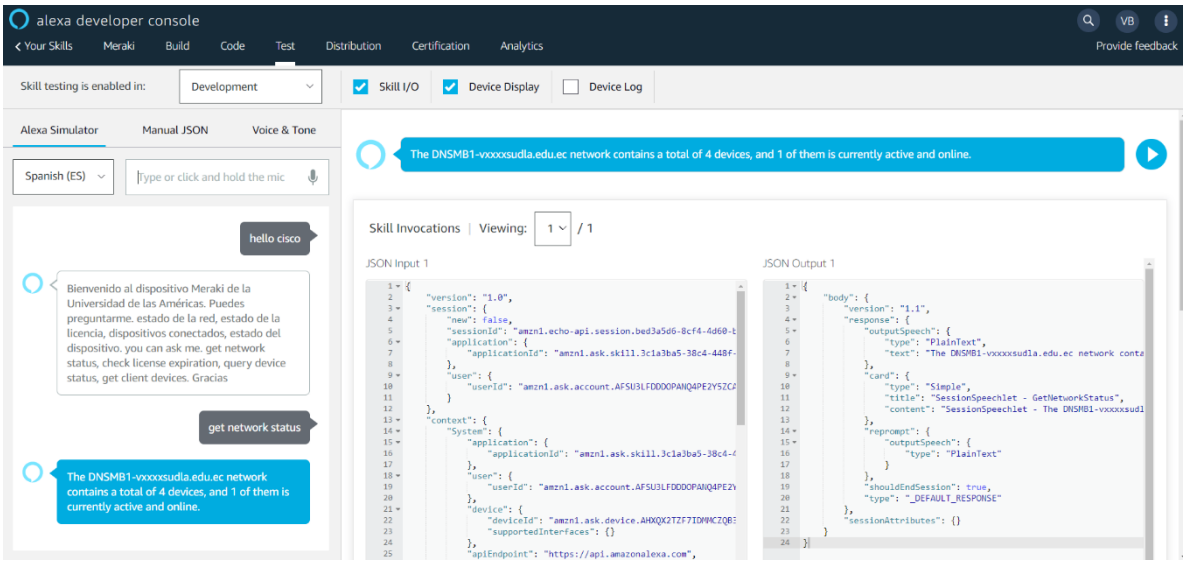


Figura 60. Estado de red.

En la figura 61 se solicita al Meraki que revise el estado de la licencia que tiene.

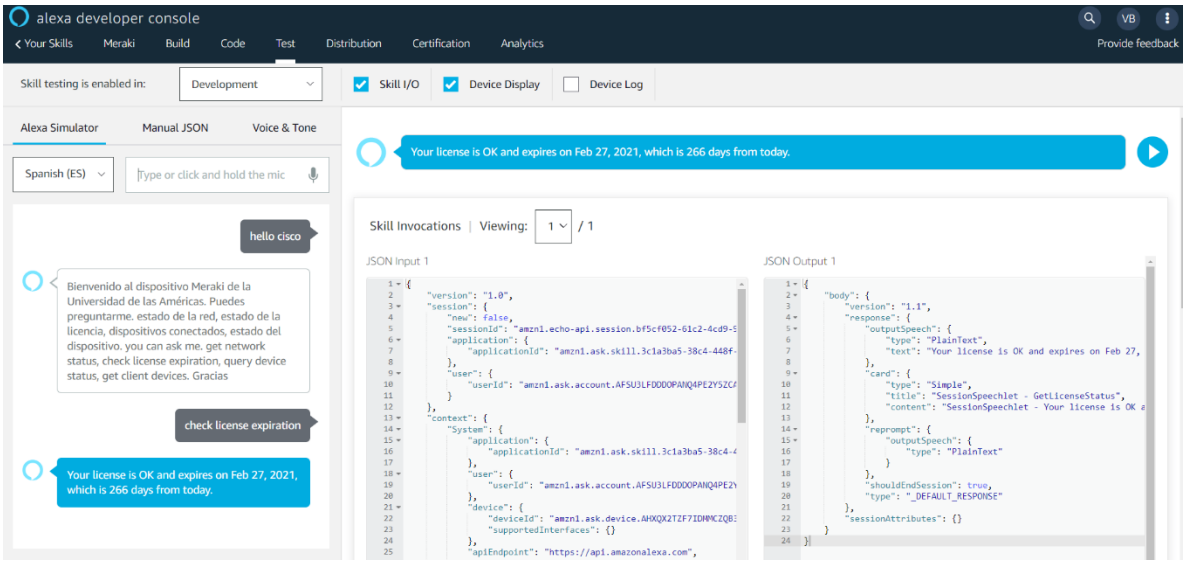


Figura 61. Licencia del dispositivo.

Otra de las peticiones que se puede realizar es solicitar el número de clientes conectados en la última hora y los recursos que ha consumido.

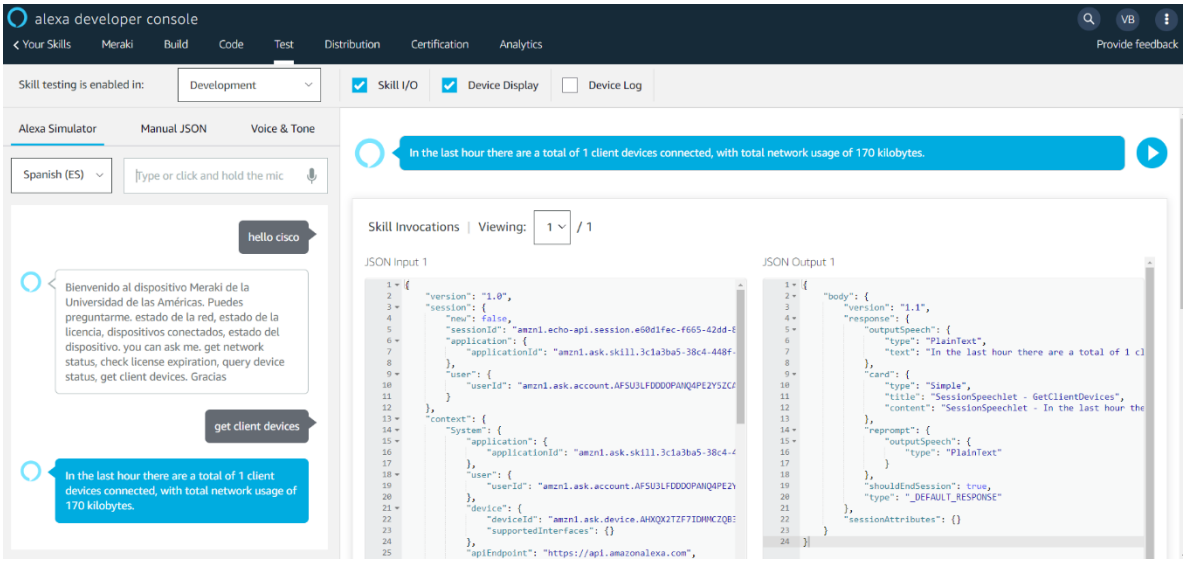


Figura 62. Clientes conectados.

6. CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

DevNet Sandbox es un simulador con ambiente controlado, este permite realizar las implementaciones de proyectos orientados a diversas ramas tales como Networking, IoT, entre otras.

Este proyecto de titulación basado en laboratorios que ofrece Cisco DevNet está orientado a personas con discapacidad, ya que brinda facilidades a los usuarios a cargo de la administración.

Por medio de la plataforma Postman se puede obtener cualquier tipo de dato o información que se necesite acerca del dispositivo Meraki, ya que Postman permite realizar consultas como por ejemplo acerca de las redes, clientes, organizaciones, entre otras.

Con la implementación del sistema para la administración de un Meraki MR33 se optimiza el tiempo de los administradores de red, debido a que pueden controlarla por medio de comandos de voz evitando tener que acceder a la plataforma e interfaz correspondiente.

Python es un lenguaje de programación de interpretación de alto nivel que trabaja por medio de scripts, los cuales ayudan a la identificación rápida de errores, además de que no impiden la ejecución del programa.

La migración del código del proyecto de administración de un UCS en el data center Académico de la Universidad de las Américas fue necesario debido a que la versión actual del código (Python 2.7) no cuenta con soporte desde febrero del 2020 por lo que es necesario realizar la migración a la versión 3.7. La exitosa migración permite un data center más accesible ya que es controlado por un asistente de voz.

Meraki MR33 cuenta con características muy novedosas, a fin de que se puede tener una red más segura donde se configura reglas de firewall para invitados, análisis en tiempo real con antivirus y administrar desde la nube.

Para la correcta administración de la red por medio de la nube se utiliza la integración de las plataformas de Amazon Web Services y Amazon Alexa, donde se configuran diversos servicios, funciones y aplicaciones para lograr gestionar un dispositivo Meraki por medio de la voz.

6.2. Recomendaciones

Antes de realizar alguna implementación es recomendable ocupar un software de simulación, para este proyecto de titulación se ocupó el simulador DevNet Sandbox que permite ejecutar las habilidades en un ambiente externo y controlado, de esa manera se puede cerciorar que al momento de la implementación no haya errores.

Es importante dar los permisos de Amazon S3 y Amazon CloudWatch Logs a la función de Amazon Web Services al momento de vincular las plataformas. Si lo permisos son los correctos se puede realizar peticiones al Amazon Echo Dot y este responderá con éxito.

El ingreso del ARN que se obtiene de la función de AWS en la plataforma de Amazon Alexa es muy importante, ya que de esa manera Alexa tendrá comunicación total con el código cargado de Python.

Es conveniente ocupar el asistente virtual de Amazon Alexa para la implementación, debido a que se trabaja con las plataformas de Amazon. Por lo cual al ser del mismo fabricante facilita la configuración que al hacerlo con otros asistentes como Google Home.

Hay que tomar en cuenta que la creación y activación de una cuenta en la plataforma Amazon Web Services (AWS) toma 24 horas en realizarse, por lo cual antes de empezar a utilizar las plataformas es necesario registrarse con anterioridad.

Si existen errores al momento de realizar la simulación, se recomienda copiar el código Json de Alexa Developer Console y ejecutarlo en la función de AWS donde se podrá visualizar si hay error en el código de Python.

Como buena práctica, es recomendable identificar y guardar el API Key del Meraki debido a que si no se añade en el código de Python no se puede realizar una

conexión con el dispositivo, provocando que se impida la comunicación con el asistente virtual.

Verificar que los datos ingresados en el archivo de Python (meraki_info), tales como Id de la organización, Id de la red y el serial sean los del dispositivo dado que si son incorrectos no se tendrá respuesta de la petición solicitada.

REFERENCIAS

Aranda, Á. (2014). Instalación y parametrización del software (UF1893) [Ebook] (1st ed., p. 259). PDF. Recuperado el 21 de abril de 2020 de https://ebookcentral.proquest.com/lib/udlasp/reader.action?docID=4310545&query=Instalaci%C3%B3n+y+parametrizaci%C3%B3n+del+software+#.

AWS | Almacenamiento de datos seguro en la nube (S3). Amazon Web Services, Inc. (2020). Recuperado el 10 de junio de 2020 de https://aws.amazon.com/es/s3/.

AWS Lambda – Serverless Compute - Amazon Web Services. Amazon Web Services, Inc. (2020). Recuperado el 10 de junio de 2020 de https://aws.amazon.com/lambda/.

Build software better, together. GitHub. (2020). Recuperado el 10 de junio de 2020 de https://github.com/.

Cisco DevNet: APIs, SDKs, Sandbox, and Community for Cisco Developers. Developer.cisco.com. (2020). Recuperado el 10 de junio de 2020 de https://developer.cisco.com/.

Cisco DevNet: APIs, SDKs, Sandbox, and Community for Cisco Developers. Developer.cisco.com. (2020). Recuperado el 10 de junio de 2020 de https://developer.cisco.com/docs/sandbox/#!getting-started/reservation-sandboxes.

Cisco DevNet Learning Labs. Developer.cisco.com. (2020). Recuperado el 10 de junio de 2020 de https://developer.cisco.com/learning/lab/Meraki_Alexa/step/1.

Cisco Meraki - Create with the Meraki Platform. Developer.cisco.com. (2020). Recuperado el 10 de junio de 2020 de https://developer.cisco.com/meraki/build/meraki-postman-collection-getting-started/.

Developer.amazon.com. (2020). Recuperado el 10 de junio de 2020 de https://developer.amazon.com/es-ES/alexa/alexa-skills-kit.>.

Developer Amazon. Developer.amazon.com. (2020). Recuperado el 10 de junio de 2020 de https://developer.amazon.com/es-ES/alexa/alexa-skills-kit#:~:text=El%20Alexa%20Skills%20Kit%20(ASK,y%20llegar%20a%20los%20consumidores.

Gallardo Vázquez, S. (2019). Configuración de instalaciones domóticas y automáticas [Ebook] (2ª ed., Pp. 109-110). Madrid. Recuperado el 10 de junio de 2020 de https://books.google.com.ec/books?id=iD2dDwAAQBAJ&pg=PA110&dq=amazon+echo+alexa&hl=es&sa=X&ved=0ahUKEwiK4pu4xbTjAhWTGc0KHQ9XD9cQ6AEIMDAC#v=onepage&q=amazon%20echo%20alexa&f=false

Haack, W., Severance, M., Wallace, M., & Wohlwend, J. (2017). Security Analysis of the Amazon Echo. Recuperado el 10 de junio de 2020 de https://pdfs.semanticscholar.org/35c8/47d63db1dd2c8cf36a3a8c3444cdeee605e4.pdf

Kalb, I. (2016). Learn to program with Python [Ebook] (1st ed., pp. 1-4). 2016. Recuperado el 10 de junio de 2020 de https://link-springer-com.bibliotecavirtual.udla.edu.ec/content/pdf/10.1007%2F978-1-4842-2172-3.pdf.

Marzal, A., Gracia, I., & García, P. (2014). Introducción a la programación con Python 3 [Ebook] (1st ed.). Universitat Jaume I. Servei de Comunicació i Publicacions. Recuperado el 10 de junio de 2020 de https://ebookcentral.proquest.com/lib/udlasp/reader.action?docID=4499415&query=Introducci%C3%B3n+a+la+programaci%C3%B3n+con+Python+3.

Meraki MR33 | Access Point | 802.11 AC Wireless. Cisco Meraki. (2020). Recuperado el 10 de junio de 2020 de https://meraki.cisco.com/products/wireless/mr33#tech-specs.

Meraki Dashboard API. Meraki Dashboard API. (2020). Recuperado el 10 de junio de 2020 de https://documenter.getpostman.com/view/7928889/SVmsVg6K?version=latest.

Meraki and Cisco Collaboration Teleworker Solution for Businesses. Cisco Meraki. (2020). Recuperado el 10 de junio de 2020 de https://documentation.meraki.com/Architectures_and_Best_Practices/Meraki_and_Cisco_Collaboration_Teleworker_Solution_for_Businesses.

Postman. Postman.com. (2020). Recuperado el 10 de junio de 2020 de https://www.postman.com/.

Sarasa, A. (2017). Gestión de la información web usando Python [Ebook] (1st ed., p. 10). PDF. Recuperado el 10 de junio de 2020 de https://ebookcentral.proquest.com/lib/udlasp/reader.action?docID=4849783&query=Gesti%C3%B3n+de+la+informaci%C3%B3n+web+usando+Python#.

Setting up Alexa for Business with Cisco Telepresence video conferencing | Amazon Web Services. Amazon Web Services. (2019). Recuperado el 10 de junio de 2020 de https://aws.amazon.com/es/blogs/business-productivity/setting-up-alexa-for-business-with-cisco-telepresence-video-conferencing/.

Universidad de las Américas. (2020). http://dspace.udla.edu.ec/bitstream/33000/12173/1/UDLA-EC-TIERI-2020-04.pdf (Ingeniero).

What's New In Python 3.7 — Python 3.7.7 documentation. Docs.python.org. (2020). Recuperado el 10 de junio de 2020 de https://docs.python.org/3.7/whatsnew/3.7.html.