DESARROLLO DE UNA APLICACIÓN UTILIZANDO AWS LAMBDA

Juan Sebastian Arias Llerena; Andres Escobar Castaño; Manuel Antonio Tarazona; David Aristizabal;
Facultad de Ingeniería, Departamento de Electrónica y Automática.
Universidad Autónoma de Occidente
Cali, Colombia

***Abstract* —En este documento se verá expuesta toda la información acerca de una aplicación de mediana complejidad utilizando AWS Lambda, se llevará a cabo todo el proceso de construcción, diseño e implementación de la aplicación, para ello se ven involucrados diversas temáticas o conceptos utilizados y afianzados a lo largo de computación en la nube. Además de explicar y describir la tecnología utilizada la cual es propia de Amazon Web Services y donde AWS Lambda es una plataforma de computación sin servidor, basada en eventos.**

Palabras clave – Serverless, Services, AWS, Amazon, Aplicación, Computación en la nube.

INTRODUCCIÓN

Antes de iniciar con el desarrollo de una aplicación, se deben tener las bases teóricas, para conocer el funcionamiento de la herramienta a utilizar para la implementación de la aplicación. La aplicación a llevar a cabo es el reconocimiento biométrico, basado en una imagen almacenada en una base de datos. El funcionamiento de esta a rasgos simples es la comparación de la imagen tomada por la cámara con la imagen almacenada en la base de datos, donde dentro de las variables de la base de datos se encuentran, el documento, un correo electrónico y una vez realizada la comparación un porcentaje de similitud entre las imágenes. Basado en ámbito real o profesional, este aplicativo tiene mucha escalabilidad, ya que puede ser implementado para ingresos a empresas y/o entidad que requiera algún tipo de seguridad al ingreso. Para llevar a cabo dicho desarrollo de aplicación se utilizará la plataforma de servicios que nos ofrece Amazon mediante el apartado de AWS educate y donde la herramienta principal es la de AWS Lambda la cual es un servicio de informática que está basado en eventos los cuales permiten ejecutar algún tipo de código y donde ofrece diversos lenguajes de programación para así ejecutar el código o aplicación sin necesidad de realizar aprovisionamiento o administrar servidores.

I. OBJETIVOS

A. General

Realizar e implementar una aplicación la cual mediante la toma de una foto realice una comparación biométrica donde si la similitud está por encima del 95% almacene los datos relacionados con este ingreso.

B. Específicos

- Utilizar la herramienta ofrecida por Amazon Web Services cómo lo es AWS Lambda.
- Utilizar base de datos, donde la designada o escogida para el desarrollo de esta aplicación es Mysql.
- Utilización de un gateway para lanzar la página web del aplicativo.

II. MARCO TEÓRICO

.

Antes de iniciar con el desarrollo de la aplicación se debe inicialmente indagar sobre las herramientas o servicios a utilizar para llevar a cabo dicha aplicación. Teniendo en cuenta que el requerimiento principal para el desarrollo de dicho aplicativo es la utilización de AWS Lambda, se inicia describiendo su funcionalidad, para que casos en especiales utilizar este servicio informático y algunos puntos claves. Pero antes de ello debemos indagar qué otras plataformas ofrecen un servicio similar donde no haya necesidad de realizar aprovisionamientos o administración de servidores. Se mencionan cinco de ellos y se profundizará mucho más en la herramienta utilizada para este proyecto cómo lo es AWS Lambda.

La primera herramienta que se describirá es Microsoft Azure Functions [1], la cual está designada cómo una de las mejores herramientas en cuanto serverless computing ya que nos permite una gran mejora y cumple con conceptos claves en dichos servicios cómo lo es la disponibilidad y la accesibilidad, esta ofrece al usuario un gran campo de personalización en cuanto a manejo y seguridad del propio aplicativo, las funciones más destacadas de dicho servicio son: completo acceso para administrar, esta función va enfocada netamente a la seguridad del aplicativo, un sistema de señales o alertas el cual permite analizar el rendimiento del sistema, así entendimiento o conocimiento de las fallas del aplicativo, permite un desarrollo simplificado de la aplicación ya que permite desarrollar aplicaciones complejas con alta eficiencia y por último permite la integración con otros servicios, ya al ser proveedor de Microsoft permite integrar.

La segunda herramienta es la Google Cloud Functions el cual es una de las más extensas firmas cuando hablamos de computación ya al ser de esta gran plataforma cómo lo es Google,

está herramienta es simple a la hora de utilizar y a la hora de personalizar según sea lo deseado, algunas de las características o funciones de esta herramienta son; la primera de ella es que es auto escalable, otra funcionalidad es que puede desplegar el código remotamente y se pueden realizar cambios cuantas veces sea necesario, al igual que permite realizar algunos cambios de seguridad con facilidad y utilizar otros dispositivos cómo el teléfono celular a través de Google Assistant y otros servicios móviles, soporta diferentes lenguajes de programación y la disponibilidad al ser de Google provee gran despliegue en diferentes alrededor del mundo donde esté presente Google.

El tercer servicio es IBM Cloud Functions el cual es un servicio que provee una versión estable y fácil en cuanto a escalabilidad de serverless computing, al igual que la anteriores puede ser implementadas con otras herramientas de IBM, entre sus funciones está que es un ecosistema el cual trabaja con Apache donde se puede editar y subir el código necesario para llevar a cabo la aplicación, permite el análisis cognitivo gracias a la herramienta que tiene integrada y es llamada cómo IBM Watson APIs la cual permite realizar alertas y administrar sistemas a través de la plataforma uniforme, por último está herramienta tiene buena escalabilidad, alta disponibilidad y que es simple al ejecutar.

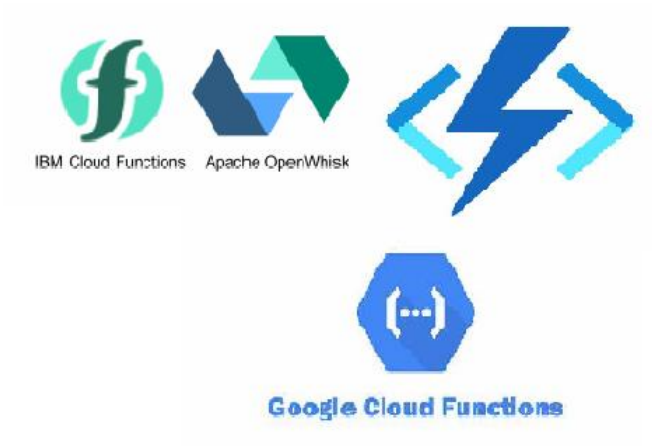

Figura 1. Herramientas de serverless computing.

Ahora si iniciaremos con la descripción principal la cual es la herramienta o servicio informático llamado AWS Lambda[2], cómo es bien sabido por la documentación de dicho servicio, está nos permite ejecutar código sin aprovisionar ni hacer uso de servidores, se conoce que existen cómo tres tipos de servicio donde Lambda es un software cómo servicio (SaaS), al igual que las anteriores herramientas estas pueden ser combinados con otros servicios ofertados por AWS tales cómo almacenamiento, gateways entre otros servicios que nos ofrece está plataforma, se dice que hay unos casos muy específicos o especiales para utilizar lambda, ya que este es ideal para aplicaciones, se describe a sí mismo cómo un servicio donde solo nos debemos preocupar por la implementación de código que al igual que las anteriores herramientas mencionadas se puede utilizar diversos lenguajes de programación cómo node js, python, entre otros[3], [4], un aspecto muy importante de Lambda es que nos ofrece una administración muy completa ya que nos otorga una combinación equilibrada entre memoria y otros recursos necesarios para el desarrollo o ejecución del código[5]. Está servicio de AWS Lambda fue la primera herramienta de serverless computing, donde algunas de sus más destacables características son; permite el control de concurrencia y escalabilidad, esto hace referencia a que nos permite realizar un control detallado sobre está escalamiento y un tiempo de respuesta óptimo de las aplicaciones en desarrollo, otra característica es que permite utilizar imágenes de contenedores, donde se realizan dependencia para así poder implementar funciones dentro de ella, también nos otorga proyectos de funciones el cual es cómo un código de ejemplo la cual nos permite conocer cómo usar está herramienta y cómo conectarlas con otros servicios ofrecidos por está plataforma, cómo sabemos cada día son más importantes los datos y este servicio no se queda atrás ya que nos permite administrar un proxy de base de datos donde se pueden realizar consultas basadas en funciones realizadas en dicho entorno, lo que nos permite que dichas funciones tengan un alto nivel de simultaneidad sin limite de conexiones a bases de datos, por último y no menos importante nos permite un acceso a sistemas de archivos donde podemos configurar una función para así montar un sistema de archivos con un directorio local, donde el código de la función puede acceder y modificar todos los recursos compartidos de una manera segura y con alta concurrencia.

Figura 2. Herramientas de serverless computing AWS Lambda.

III. ANÁLISIS Y RESULTADOS

Según el objetivo trazado para este proyecto final y basado en los requerimientos específico tales cómo el uso de un AWS Lambda, se plantea un aplicativo cuyo funcionamiento sea el reconocimiento biométrico para el login o registro basado en una comparación con una imagen previamente almacenada, a

grandes rasgos lo que debe realizar dicho aplicativo es la comparación biométrica de la imagen almacenada en la plataforma de S3 con una toma de fotografía en una página web, donde quedará almacenada dicha información si la similitud está por encima del 95%.

Se piensa en dicho aplicativo ya que a lo largo del tiempo nos hemos envuelto en la problemática de falsificación de ingresos a empresas, industrias entre otros entes, donde la suplantación y otras faltas se han vuelto muy comunes y donde la seguridad se ha cumple un rol muy importante para nuestra sociedad, es por esto que se plantea este aplicativo, para mitigar dichas violaciones a la seguridad de manera total, ya que nos basamos en un modelo muy específico ya que es un sistema muy privado, se tienen una base de datos principal donde se verán almacenados los datos más importantes según la propuesta planteada.

El nivel de escalabilidad de dicho aplicativo es muy grande, ya que nos permite el modelado e ingreso de diferentes comparaciones biométricas en donde el software o algoritmo de identificación arroja resultados óptimos en cuanto a comparación se refiere, esto nos da a entender que se tiene con un algoritmo robusto y confiable donde se pueden realizar diferentes adaptaciones y mejoras para que sea aún más confiable. Se iniciará el pasó a pasó de la implementación de dicho aplicativo con el fin de demostrar el uso de una herramienta cómo lo es AWS Lambda y cómo con esta se pueden realizar aplicativos con alta escalabilidad, disponibilidad y con un amplio margen de mejoramiento.

IV. FUNCIONAMIENTO

Para la implementación se utilizó AWS lambda y se crea un almacenamiento en S3 que es otro servicio de AWS, sirviendo para subir las imágenes, además, se hizo uso de dos bases de datos, una en DynamoDB[6] que almacenará todas las sesiones y otra en MySQL que sólo almacenará las sesiones que cumplan con un porcentaje de similitud superior a un 95% cómo se muestra en la figura 3. Luego se utiliza otra función de AWS llamada API Gateway[7] la cual permite generar un enlace de acceso a la aplicación en cualquiera de la etapas de la implementación.

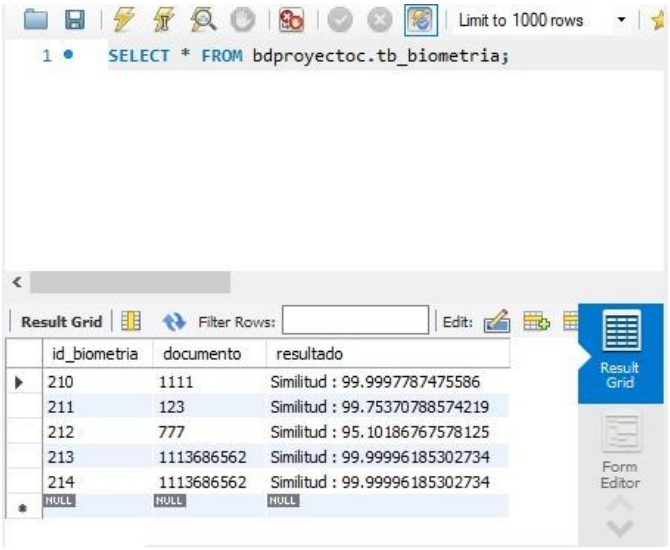

Figura 3. Base de datos en MySQL

Se usaron dos funciones que están conectadas entre ellas y están desarrolladas en dos diferentes entornos de ejecución, la primera es una función lambda biometría [8] que permite realizar el reconocimiento facial mediante la cámara del dispositivo de cómputo y comparar con una imagen ha sido almacenada con anterioridad.

Utilizando AWS rekognition[9] el cual permite identificar distintos parámetros para realizar comparaciones entre imágenes y estas son enviadas a la bases de datos en MySQL (Ver Anexo 2) si cumplen con el requisito impuesto. Para la segunda función[10] se tienen dos paginas web y una función principal en Python que permitir decidir en caso de GET y/o POST y a que página redirigir (Ver Anexo 1). La principal permitirá realizar un request con el Gateway que le permitirá conectarse y con un método post redirigirá a otra página que informa sobre la finalización de la solicitud y en caso de cumplir con los requisitos podrá realizar un login o cualquier otra operación.

Figura 4. Ejemplo de pantalla de inicio

Cómo se puede apreciar en la implementación mostrada en la figura 4 la interfaz consta de cuatro componentes que son: ingreso de documento, captura de una imagen con la cámara del dispositivo, un email y finalmente un botón de enviar. Una vez ingresados estos datos, estos se registrarán en las bases de datos antes mencionadas y lanzando en una pantalla que informa de la finalización de la operación como en la figura 5.

Figura 5. Página de finalización de captura de datos

V. CONCLUSIONES

- Para el desarrollo de este aplicativo se indagó en las distintas alternativas para realizar implementaciones de funciones de cualquier índole haciendo uso de serverless, tales como Azure functions, Google cloud y AWS lambda, averiguando sobre las posibilidades y funcionamiento de estas y sus beneficios de usar estas tecnologías.
- Se decidió utilizar AWS lambda debido a la cantidad de información disponible en la web sobre este servicio, además, de códigos y herramientas disponibles y listas para usar que AWS tiene a disposición como lo fue el caso de AWS Rekognition y finalmente, se eligió por ser uno de los servicios de serverless más utilizados.

REFERENCIAS

[1] "Top 5 Serverless Computing Tools | Knowldgenile". https://www.knowledgenile.com/blogs/serverless-computing-tools/ (consultado nov. 09, 2021).

[2] "AWS Lambda – Características del producto", *Amazon Web Services, Inc.* https://aws.amazon.com/es/lambda/features/ (consultado nov. 09, 2021).

[3] "🐑 QUE ES AWS LAMBDA? INTRODUCCION EN ESPAÑOL A SERVERLESS EN AWS 🐑 - YouTube". https://www.youtube.com/watch?v=1wNb_RMvI9E (consultado nov. 09, 2021).

[4] miguel rodriguez, *AWS Lambda Node JS , API Rest y RDS Mysql ejemplo CRUD*, (sep. 21, 2021). Consultado: nov. 09, 2021. [En línea Video]. Disponible en: https://www.youtube.com/watch?v=da6E0R2grfo

[5] "Introducción al Lambda - AWS Lambda". https://docs.aws.amazon.com/es_es/lambda/latest/dg/getting-started.html (consultado nov. 09, 2021).

[6] "¿Qué es Amazon DynamoDB? - Amazon DynamoDB". https://docs.aws.amazon.com/es_es/amazondynamodb/latest/developerguide/Introduction.html (consultado nov. 09, 2021).

[7] "Amazon API Gateway | API Management | Amazon Web Services", *Amazon Web Services, Inc.* https://aws.amazon.com/es/api-gateway/ (consultado nov. 09, 2021).

[8] miguel rodriguez, *AWS Lambda Node JS , Rekognition , RDS Mysql ejemplo Validación Biométrica*, (sep. 21, 2021). Consultado: nov. 09, 2021. [En línea Video]. Disponible en: https://www.youtube.com/watch?v=ylbOssrqlco

[9] "Análisis de imágenes y videos con machine learning - Amazon Rekognition - Amazon Web Services", *Amazon Web Services, Inc.* https://aws.amazon.com/es/rekognition/ (consultado nov. 09, 2021).

[10] "AWS Dojo - Workshop - Deploy Simple Web Applications in AWS Lambda". https://aws-dojo.com/ws47/labs/create-lambda-function/ (consultado nov. 09, 2021).

ANEXOS

```
import json
import os, re, base64
import boto3

def lambda_handler(event, context):

    mypage =
page_router(event['httpMethod'],event['queryStringParame
ters'],event['body'])

    return mypage


def page_router(httpmethod,querystring,formbody):

    if httpmethod == 'GET':
        htmlFile = open('contactus.html', 'r')
        htmlContent = htmlFile.read()
        return {
        'statusCode': 200,
        'headers': {"Content-Type":"text/html"},
        'body': htmlContent
        }

    if httpmethod == 'POST':

        insert_record(formbody)

        htmlFile = open('confirm.php', 'r')
        htmlContent = htmlFile.read()
        return {
        'statusCode': 200,
        'headers': {"Content-Type":"text/html"},
        'body': htmlContent
        }

def insert_record(formbody):
```

```
        formbody = formbody.replace("=", "' : "')
        formbody = formbody.replace("&", "', "')
        formbody = "INSERT INTO dojotable value {"' +
    formbody +  "'}"

        client = boto3.client('dynamodb')
 client.execute_statement(Statement= formbody)
```

Anexo 1. Código de redirección de páginas

```
const AWS = require('aws-sdk');
const mysql = require('mysql');
const con = mysql.createConnection({
  host: 'database-1.c5vhhsc8kyrb.us-east-2.rds.amazonaws.com',
  user: 'admin',
  port:"3306",
  password: 'proyectoc',
  database: 'bdproyectoc',
});


exports.handler = async (event) => {

        // OBTENER FOTO S3 y lo PONEMOS EN BUFFER
        let Bucket = "ue2stglusoft2000/imagen";
        let Key = "Aristi.jpeg.jpeg";
        let requestBUCKET = {
          Bucket,
          Key,
        };
        let s3bucket = new AWS.S3();
        let fotografia = await
s3bucket.getObject(requestBUCKET).promise();
        const bufferFoto = await
Buffer.from(fotografia.Body.toString('base64').replace(/^data:image\/\
w+;base64,/, ""), "base64");
        // OBTENER FOTO POST y lo PONEMOS EN BUFFER.
        const bufferValidacion = await
Buffer.from(event.imagen.replace(/^data:image\/\w+;base64,/, ""),
"base64");
        //VALIDACION FACIAL
        const params_ = {
          SimilarityThreshold: 90,
          SourceImage: { Bytes: bufferFoto},
          TargetImage: { Bytes: bufferValidacion}
        };
        var rekognition = new AWS.Rekognition();
        let compareFacesResponse = await
rekognition.compareFaces(params_).promise();
        let resultado="Similitud :
"+compareFacesResponse.FaceMatches[0].Similarity;

        // REGISTRAMOS EN LA BASE DATOS
         let sql = "INSERT INTO tb_biometria (documento,resultado)
VALUES ("'+event.documento+'","'+resultado+'")";
         await new Promise((resolve, reject) => {
           con.query(sql, (err, res) => {
                if (err) {
                  throw err
                }
           resolve("OK");

           });
         })
       const response = {
       statusCode: 200,
       body: JSON.stringify("Se registro BD y tu "+ resultado),
       };
       return response;

  };
```

Anexo 2. Código función de reconocimiento facial