Implementación serverless de un sistema de reconocimiento facial para controlar el acceso

Alejandro Martín Simón Sánchez
MUii
Universidad de Castilla-La Mancha
Albacete, España
alejandromartin.simon@alu.uclm.es

Carlos Negrillo Pastor
MUii
Universidad de Castilla-La Mancha
Tobarra (Albacete), España
carlos.negrillo@alu.uclm.es

Antonio Morán Muñoz
MUii
Universidad de Castilla-La Mancha
Albacete, España
antonio.moran1@alu.uclm.es

Gabriel Gómez López
MUii
Universidad de Castilla-La Mancha
Lezuza (Albacete), España
gabriel.gomez3@alu.uclm.es

***Resumen*—La inversión de recursos software y hardware dedicados a la implementación de un sistema escalable y altamente disponible tiene una gran importancia para proveer un servicio correcto de forma ininterrumpida. Un sistema debe estar ajustado en las condiciones adecuadas a la demanda exigida. Es por ello que se propone el ejemplo de la implementación serverless de un sistema de control de acceso a edificios mediante reconocimiento facial. La implementación se lleva a cabo utilizando herramientas como el framework serverless para desplegar funciones lambda que se ocuparán del procesamiento, espacios de almacenamiento utilizando buckets de S3, y una base de datos DynamoDB para almacenar la información relacionada con el proyecto. Así también se contempla el uso del servicio S3 para alojar una aplicación web. De esta forma se obtiene un servicio ajustado a las necesidades con un sistema que escala y en el que se invierten recursos económicos de acuerdo al uso real.**

***Index Terms*—Reconocimiento facial, Control acceso, AWS, DynamoDB, Balanceo de carga, API, S3, boto3, Escalabilidad.**

I. Introducción

El control de acceso es una de las funciones más importantes a la hora de hacer seguro un sistema o unas instalaciones. Esto tiene aún más importancia si cabe en centros públicos como las universidades, donde hay gran trasiego de estudiantes y es importante asegurar la seguridad de los mismos [1]. No obstante, los sistemas tradicionales suelen ser lo suficientemente caros como para que su adopción por parte de instituciones públicas como las universidades resulte inviable. Con ello en mente, se propone en el presente documento una propuesta de control de acceso montada sobre la infraestructura del proveedor Cloud público AWS. La toma de fotos estaba pensada hacerse mediante Raspberry Pi en un principio, pero por circunstancias excepcionales no se ha podido tener acceso a dichos dispositivos. Por lo tanto, se ha sustituido dicha opción por una web a través de la que se pueden subir fotos de usuarios que van entrando. El código fuente de este proyecto es accesible desde https://github.com/anmomu/practica-abp-pygitic.

II. Estado de la cuestión

La computación Cloud es un negocio en auge desde ya algunos años, pues es una solución para muchas organizaciones que no tienen ni el personal ni los medios económicos necesarios para montarse una infraestructura propia sobre la cual llevar el negocio. Los servidores Cloud públicos actuales ofrecen una serie de servicios que se despliegan sobre su propia infraestructura y que tienen la característica de ser muy escalables y poder ser ofrecidos bajo demanda, por lo que se pueden eliminar o aprovisionar recursos en función de las necesidades particulares. Para la implementación de este sistema de reconocimiento facial y control de acceso se hace uso de la infraestructura ofrecida por Amazon, *Amazon Web Services* o *AWS*. Como todo servidor Cloud público, AWS ofrece toda una gama de servicios a través de su portal web. A continuación, se describirán los servicios necesarios para montar nuestro sistema.

II-A. Amazon Boto3

Boto es el entorno de desarrollo software (Software deployment kit, SDK), de AWS en Python. Permite a los desarrolladores crear, configurar y gestionar servicios de AWS, tales como EC2 y S3. Boto proporciona una forma sencilla de utilizar la programación orientada a objetos y un acceso de bajo nivel a los servicios de AWS [2].

II-B. Amazon S3

S3 es la abreviatura de *Amazon Simple Storage Services*. S3 es un servicio de almacenamiento de objetos que ofreces escalabilidad, disponibilidad de datos, seguridad y rendimiento. Esto permite que clientes de diferentes tamaños y sectores puedan utilizarlo, ya sea para almacenar datos de sitios web, aplicaciones móviles, procesos de copia de seguridad o análisis de bid data. Permite administrar y monitorizar el almacenamiento, realizar consultas “in situ” y llevar a cabo transferencias de datos. [3]

II-C. *Amazon Rekognition*

Amazon Rekognition es un servicio de AWS que permite el análisis de imagen y vídeo. Dicho servicio lleva a cabo algoritmos de aprendizaje automático ya implementados, por lo que no se requiere de experiencia en dicho campo para su manejo. Una de las características de Amazon Rekognition es que proporciona la capacidad de detectar, analizar y comparar rostros, por lo que tiene especial utilidad en casos de uso que incluya verificación de usuarios o control de acceso. Otra de las características es que este servicio dispone de una API a través de la cual se puede acceder. Además, también permite el acceso a través de la consola de administración de AWS y a través de la linea de comandos de AWS. [4]

II-D. *Amazon DynamoDB*

Amazon DynamoDB es una base de datos de clave-valor y documentos que ofrece una latencia de milisegundos de un solo dígito a cualquier escala. También puede gestionar más de 10 billones de solicitudes por día (admitiendo picos de 20 millones de solicitudes por segundo). Además, dispone de copia de seguridad, restauración y seguridad integradas. Otras características que son importantes de cara al control de acceso son las siguientes:

- Replicación global automatizada con tablas globales: esto permite un rápido acceso a los datos independientemente de la aplicación, así como el aumento de la seguridad de los mismos en caso de que falle una instancia.
- Escalado automático: los recursos se van aprovisionando a medida que se van necesitando. Tampoco es necesario aprovisionar y gestionar un servidor sobre el que montar la base de datos, ya que se lleva a cabo automáticamente.

II-E. *Amazon Lambda*

Amazon Lambda es el servicio de AWS que permite ejecutar código sin necesidad de crear ni administrar servidores o intancias EC2, pagando sólo el tiempo de cómputo que consuma las ejecuciones de las lambdas creadas. Se puede configurar el código para que se active automáticamente desde otros servicios de AWS o se puede llamar directamente desde cualquier aplicación web o móvil. Un punto a favor de usar lambdas es su capacidad de escalar automáticamente y la alta disponibilidad (hasta 1000 ejecuciones por segundo) [5].

III. Arquitectura

En esta sección se describe la relación entre los diferentes elementos que componen el sistema. Para ello, se describirán los flujos de datos entre dichos elementos, así como la secuencia de pasos de las que consta la aplicación. En la Figura 1 se muestra una visión general de la arquitectura del sistema de control de acceso y reconocimiento facial, donde se pueden ver los diferentes elementos que lo componen y las conexiones que existen entre ellos.

Algo que destaca en la arquitectura es la ausencia de cámaras que hagan fotos a los individuos que entren a la instalación. Como ya se ha indicado, esto se debe a la imposibilidad de tener acceso a las Raspberry Pi con cámaras. Por ello, las imágenes se subirán manualmente a través del navegador, donde se mostrará si dicho individuo tiene permisos de acceso o no.

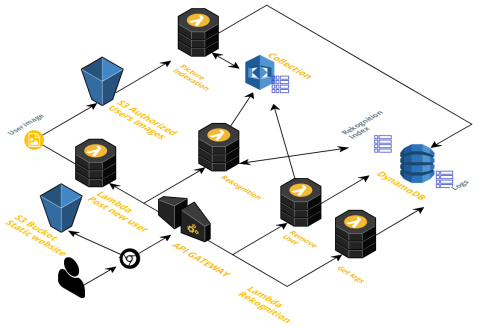

Figura 1. Arquitectura general del sistema

A continuación, se pasa a describir uno por uno los diferentes subsistemas que conforman la infraestructura, dentro de los cuales se describen por orden cada uno de los flujos y pasos que se llevan a cabo.

III-A. *Indexación facial usando Rekognition (Gestión de usuarios)*

Se trata del punto de partida del sistema, ya que es el flujo de la arquitectura que se encarga de introducir nuevos usuarios al sistema de personas permitidas. Este sistemas de personas permitidas, consta de 2 elementos de almacenamiento:

- Rekognition Collection: se trata de un almacenamiento clave-valor, accesible mediante llamadas al API de Rekognition. En esta collection guardaremos como valor, la meta-información de las imágenes que queremos que tengan acceso al edificio.
- DynamoDB Table “Rekognition-Index”: una tabla que nos permita almacenar información, más allá de la meta-información de la cara, como puede ser nombre, apellidos, o nivel de permisos dado el caso.

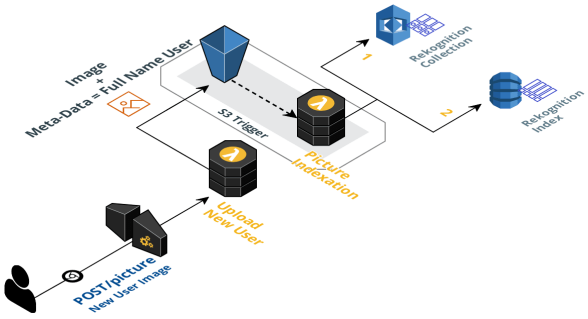

Figura 2. Flujo para añadir usuarios

Como se puede observar en la Figura 2, este flujo se inicia cada de vez que nuestra API detecta una petición POST, solicitando añadir una imagen a nuestro bucket S3. Una función lambda disparada mediante eventos API, se encarga de añadir la imagen al bucket, incluyendo el nombre y apellidos, entre los meta-datos.

Añadir esta imagen a S3, desencadenará el inicio de otra función lambda "Picture Indexation". Esta se encarga de realizar dos tareas que tienen que ver con los 2 elementos de almacenamiento mencionados previamente.

- En primer lugar, almacena en Rekognition Collection la meta-información resultante de analizar la cara de la imagen añadida a S3 previamente.
- En segundo lugar, se almacena en DynamoDB el resto de información referente al usuario, utilizando como clave principal, la misma clave creada anteriormente en Rekognition Collection.

Esto nos permitirá obtener la integración entre las dos tecnologías, y las ventajas de cada una de ellas, teniendo de este modo un usuario correctamente registrado y preparado para futuros análisis.

De manera análoga, al proceso anterior, podremos eliminar usuarios del sistema, simplemente conociendo el nombre del usuario. Obtendremos el identificador de ese usuario en DynamoDb, para posteriormente eliminarlo de Rekognition collection, como se puede apreciar en la Figura 3.

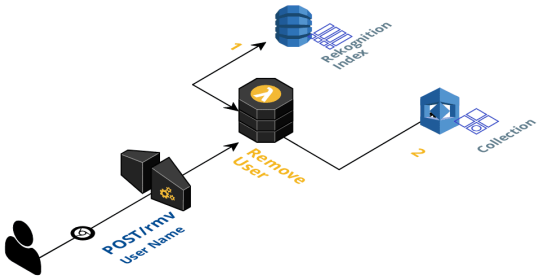

Figura 3. Flujo para eliminar Usuarios

III-B. Reconocimiento facial y registro de accesos

Este flujo de datos y procesos, es el que más veces se llegará a usar a lo largo del día, ya que es el encargado de conceder los permisos de acceso a los usuarios. Es por ello que los elementos de la arquitectura que lo conforman deben permitir escalabilidad, fiabilidad y seguridad. Gracias al uso de la API de Rekognition, funciones lambda y sencillas queries a DynamoDB conseguiremos este objetivo.

Como se puede apreciar en la Figura 4, con cada intento de acceso al edificio, nuestra API recibirá una petición POST con la imagen de la persona que quiere acceder, y lanzará una instancia de lambda que se encargará de gestionar ese acceso. La función lambda invoca la API de Rekognition para que este se encargue de comprobar si la cara de la imagen dada, se encuentra en la Rekognition Collection, con las caras de

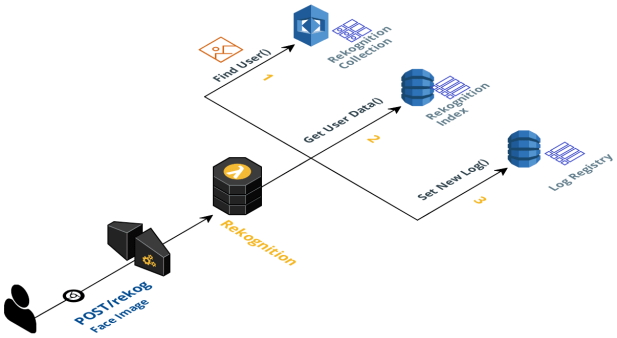

Figura 4. Arquitectura encontrar y registrar usuario

las personas autorizadas previamente. En caso de encontrarse coincidencias superiores al 95 %, el usuario tendrá concedido el acceso, y posteriormente, se creará un registro del acceso de ese usuario en la Tabla Log-Registry.

III-C. Obtención de registros de acceso

Como característica esencial de nuestro sistema, la seguridad debería ser uno de los factores más importantes. Es por ello, que para tener un control exhaustivo de los accesos, como mínimo un sistema debe dejar constancia de los accesos realizados por cada usuario, como se realiza en el flujo de la Figura 4.

Cada registro consta de la fecha de entrada, identificador y nombre de usuario, y porcentaje de acierto con el que el usuario accedió. Con una sencilla función lambda, que se encargue de realizar consultas a la Tabla Log-Registry (Figura 1), obtendremos los registros en función de los criterios deseados. En nuestro caso hemos distinguido dos claros usos de esta función:

- Últimos 5 registros de un usuario: cada vez que el usuario entre al edificio, se le mostrarán sus últimos 5 accesos, de modo que pueda vigilar posibles registros fraudulentos.
- Últimos X registros: permite al administrador obtener de manera rápida los últimos accesos de todos los usuarios, y del último día.

III-D. Sitio web usando S3

Inicialmente se había pensado en utilizar una Raspberry Pi para capturar imágenes mediante una cámara para ser procesadas mediante los servicios de AWS. Dado que por las limitaciones de la emergencia sanitaria no se puede disponer de este tipo de material para realizar el desarrollo y las pruebas, se han contemplado diversas soluciones, el uso de instancias EC2 o aumentar la funcionalidad de la aplicación web que se usaría como demo.

La aplicación web permite interactuar con el sistema. En el planteamiento inicial únicamente se requería que fuera una web de consulta para acceder a los registros de los accesos. En el planteamiento actual, considerando la situación, todas las interacciones con el sistema se realizan a través de la aplicación web para así realizar una demostración de su funcionamiento de forma elegante.

La aplicación web para acceder a los servicios del sistema y mostrar el funcionamiento se ha implementado con el framework Angular, que hace uso de TypeScript. El resultado de la compilación es alojado y ofrecido aplicación web a través de Amazon S3.

Amazon S3 ofrece la opción de alojar un sitio web estático. Un sitio web estático es aquel que se genera en el lado cliente, sin depender de código que se ejecuta en el lado servidor.

Por otro lado, un sitio web dinámico depende del procesamiento en el lado del servidor, incluidos los scripts del lado del servidor, como en PHP, JSP o ASP.NET. Amazon S3 no es compatible con páginas web cuya generación y uso depende de scripts del lado del servidor [6].

Ha consistido en la creación de dos buckets (figura 5): "accessviewer.com" y "www.accessviewer.com". En el primero se alojan los elementos correspondientes al sitio web estático, con permisos de lectura públicos y con la configuración para activar el alojamiento de sitios web estáticos. En el segundo, con el subdominio "www", se redirigen las peticiones al punto de acceso del primer bucket.

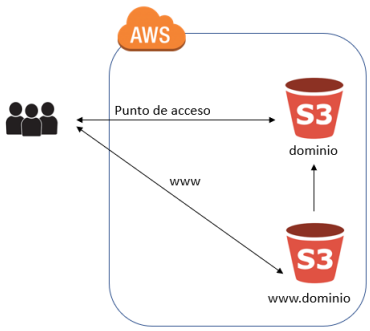

Figura 5. Sitio web estático en S3 - Arquitectura

Una aplicación en Angular sigue el patrón Modelo - Vista - Controlador, y está compuesta por componentes, que se corresponden a las vistas, en HTML y CSS, y cuya lógica está escrita en TypeScript [7]. El hecho de que se estructure en componentes facilita elaborar una aplicación con una estructura ordenada y la reutilización de código. Además ha sido implementada pensando en los usuarios que se conecten a través de diferentes plataformas, realizando una implementación del diseño responsive.

Esta aplicación realiza llamadas a la API que ha sido creada con el servicio API Gateway para hacer uso de la infraestructura desplegada en Amazon Web Services.

Está estructurada en tres partes bien diferenciadas:

- Panel de control de acceso. Está compuesto por cuatro partes:
  - Un apartado para subir la imagen. Se selecciona una imagen en formato JPEG, que es transformada a una cadena de texto en base64. Posteriormente se sube la imagen, realizando una petición post a la API con endpoint "/rekog".
  - Un visor, implementado utilizando la librería Leaflet [8]. En él se muestran unos puntos sobre el mapa donde se simula tener controles de acceso. Uno de ellos se marcará en rojo o verde según si el acceso es autorizado o denegado por el sistema de control de acceso.
  - Datos de la identificación: muestra la identidad del usuario y además el índice de confianza con el que se ha identificado a dicho usuario tras procesar la imagen.
  - Un log de accesos. Se muestra cuando el acceso ha sido autorizado, en él se muestran los últimos accesos del usuario identificado ordenados por fecha, indicando la identidad, la fecha y la hora del acceso. Está hecho con un componente reutilizable, que muestra la información que se le pasa como parámetro.
- Panel de administración. Está compuesto por tres partes:
  - Log de accesos. Muestra los últimos intentos de acceso en el sistema, tanto autorizados como denegados, ordenados por fecha. Se puede seleccionar el número de accesos que se desea mostrar. Muestra la identificación del usuario, "Unknown" en caso de ser denegado, la fecha y la hora del acceso. Se obtiene realizando una petición POST a la API con endpoint "/logs"
  - Panel de adición de imágenes. Permite añadir imágenes nuevas asociadas a un usuario, indicado en dicho panel. Muestra una confirmación cuando el dato ha sido añadido correctamente. Se efectúa realizando una petición POST a la API con endpoint "/picture"
  - Panel de eliminación de usuarios. Permite indicar el nombre de un usuario y eliminarlo para restringir su acceso. Muestra una confirmación cuando el usuario ha sido eliminado correctamente, si el usuario no estuviera presente en el sistema, mostraría un error. Se efectúa realizando una petición POST a la API con endpoint "/rmv"
- Información del grupo. Muestra información de los integrantes del grupo y del proyecto.

IV. Experimentos y resultados

En esta sección se detallan las comprobaciones mediante las cuales se comprueba el correcto funcionamiento del sistema.

El cliente web a través del cual se interacciona con el sistema es accesible desde la url http://accessviewer.com.s3-website-us-east-1.amazonaws.com.

Para mostrar el funcionamiento del sistema, lo primero sería añadir usuarios nuevos que no estuvieran registrados previamente o que se hubieran eliminado. Por ejemplo, se registran a "carlos" y "alex", en el panel de administración, seleccionado la opción de "añadir usuario". Ver figura 6.

Así, si se sube la imagen de "carlos" desde el panel de control de acceso (figura 7), el acceso estará autorizado, ofreciendo retroalimentación acerca de ello, tanto con un panel

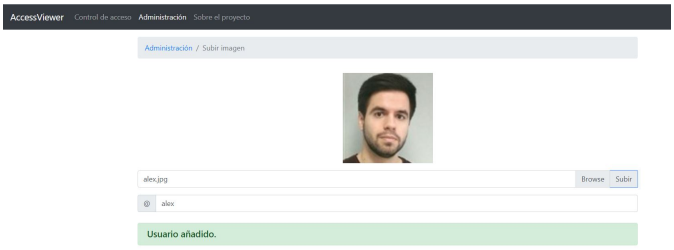

Figura 6. Aplicación Web - Añadir usuario

en color verde justo debajo de la foto, como con un indicador en el mapa del visor, donde el punto correspondiente al acceso cambia a color verde.

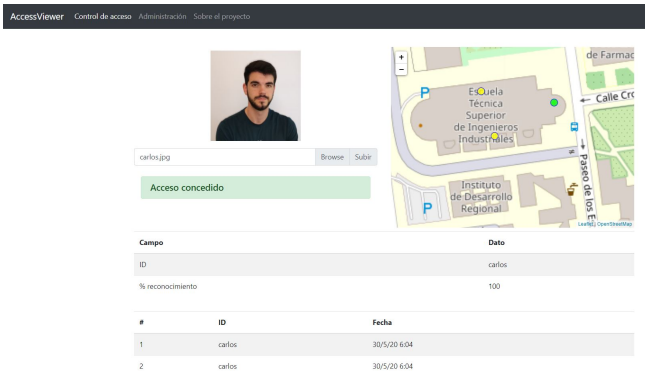

Figura 7. Aplicación Web - Control de acceso: autorizado

Los accesos se pueden comprobar desde el panel de administración (figura 8), donde aparecen los logs de los últimos accesos. El número de accesos que se muestran se puede cambiar en función del límite seleccionado. Desde el panel de administración también se accede al panel de adición de usuario (figura 6) y de eliminación de usuario (figura 9).

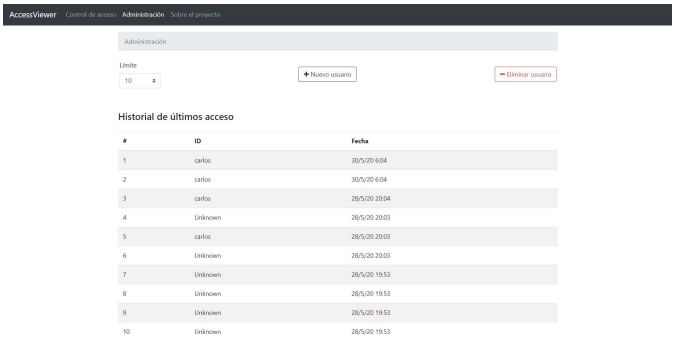

Figura 8. Aplicación Web - Logs de accesos

Desde el panel de eliminación de usuario (figura 9) se puede escribir el nombre del usuario a eliminar. En este caso, se elimina a “carlos”.

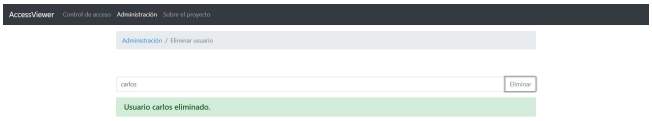

Figura 9. Aplicación Web - Eliminar usuario

Así, si se elimina al usuario “carlos” desde el panel de control de acceso (figura 10), el acceso estará denegado, ofreciendo retroalimentación acerca de ello, tanto con un panel en color rojo justo debajo de la foto, como con un indicador en el mapa del visor, donde el punto correspondiente al acceso cambia a color rojo.

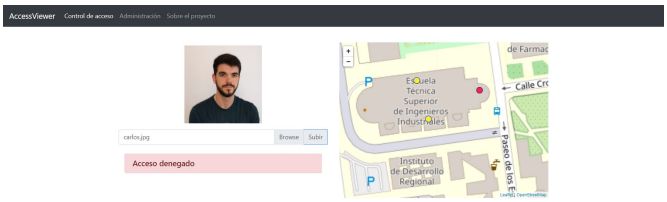

Figura 10. Aplicación Web - Control de acceso: denegado

Con la finalidad de evaluar el tiempo medio de respuesta, se ha calculado el tiempo que se tarda en recibir la respuesta del servidor. Tras realizar 10 ejecuciones el tiempo medio es de 909.9 ms, siendo un tiempo de respuesta inferior a 1 segundo para un servicio de este tipo es aceptable. Ver figura 11.

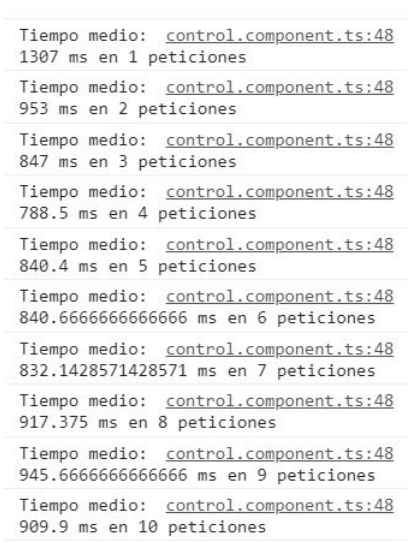

Figura 11. Servicio rekog - Tiempo respuesta

V. Conclusiones

Como se ha podido comprobar a lo largo del desarrollo del trabajo, Amazon Web Services resulta una opción de gran utilidad para organizaciones con limitada capacidad técnica o recursos económicos. La funcionalidad clave del sistema, el reconocimiento facial, ha podido ser llevada a cabo con una serie

de servicios que provee Amazon que de lo contrario habrían sido más complicados de implementar, ya que requieren de unos conocimientos avanzados de "Deep Learning".

Esta implementación también ha podido ser llevada a cabo sin contratar máquinas virtuales o instancias, resultando en una implementación totalmente "serverless". Así pues en lugar de contratar el uso de instancias EC2 junto con balanceadores de carga, se han podido utilizar funciones lambda para el procesamiento y para alojar la aplicación web, contenedores de S3, resultando en un ahorro importante de recursos económicos, ajustándose a las necesidades de este proyecto académico.

VI. Trabajo futuro

Como se ha mostrado a lo largo del documento, esta es una primera aproximación a la implementación de un sistema de control de acceso, es una demostración, por lo que quedan muchos aspectos que se pueden mejorar.

Actualmente para demostrar el funcionamiento del sistema se utiliza un panel de control que permite elegir fotografías y comprobar si se autoriza el acceso en función de si el usuario está presente en el sistema.

En el sistema real de control de acceso basado en el reconocimiento facial de los usuarios se debe implementar utilizando un dispositivo que capture vídeo o tome imágenes de los individuos que vayan intenten acceder, y las suba directamente al sistema desplegado en Amazon Web Services para que realice el procesamiento basado en reconocimiento facial.

Se pueden implementar otras formas de interacción con el sistema, ya que al ser un proyecto meramente académico, el tiempo representa un límite importante. Por ejemplo, para eliminar usuarios, se puede ofrecer una lista de usuarios que pertenecen al sistema. Así también se pueden ofrecer otras formas de visualizar los intentos de accesos, que puedan estar basados en fechas, visualizando los accesos de un determinado mes.

Aparte de optimizar el sistema, se podrían abordar una serie de mejoras encaminadas a incrementar la funcionalidad. Una de las funciones extra que se podrían añadir a este sistema sería la de detectar la temperatura corporal de los usuarios que tratan de acceder. Esta funcionalidad sería especialmente útil para controlar posibles casos de infecciones, como es especialmente interesante actualmente con la crisis sanitaria.

Agradecimientos

Gracias especialmente a Juan Ignacio Alonso Barba y Carmen Carrión Espinosa por el esfuerzo, el seguimiento y tiempo dedicado en ofrecer consejo y soporte a la hora de resolver problemas que han surgido durante el desarrollo del presente trabajo. El presente trabajo ha sido posible gracias al crédito disponible dado el carácter educativo de las cuentas utilizadas en Amazon Web Services.

Referencias

[1] La Vanguardia, "Detenido tras apuñalar a una universitaria en la Politécnica de Albacete," https://www.lavanguardia.com/sucesos/20111020/54232676450/detenido-tras-apunalar-a-una-universitaria-en-la-politecnica-de-albacete.html, [Online] 23-05-2020.

[2] Amazon Web Services, Inc., "Amazon Boto3," https://boto3.amazonaws.com/v1/documentation/api/latest/index.html, [Online] 23-05-2020.

[3] ——, "Amazon S3," https://aws.amazon.com/es/s3/, [Online] 26-05-2020.

[4] ——, "Amazon Recognition," https://aws.amazon.com/es/rekognition/, [Online] 23-05-2020.

[5] ——, "Amazon DynamoDB," https://aws.amazon.com/es/dynamodb/, [Online] 23-05-2020.

[6] ——, "Amazon S3 - Static website," https://docs.aws.amazon.com/es_es/AmazonS3/latest/dev/WebsiteHosting.html, [Online] 26-05-2020.

[7] Angular, "Angular," https://angular.io, [Online] 29-05-2020.

[8] Leaflet, "Leaflet," https://leafletjs.com, [Online] 29-05-2020.