Práctica Final ABP

Hernán Indíbil de la Cruz Calvo
Máster Universitario en Ingeniería Informática
Escuela Superior de Ingeniería Informática
Albacete, España
hernanindibil.lacruz@alu.uclm.es

Pedro Gómez López
Máster Universitario en Ingeniería Informática
Escuela Superior de Ingeniería Informática
Albacete, España
pedro.gomez5@alu.uclm.es

Alejandro Moya Moya
Máster Universitario en Ingeniería Informática
Escuela Superior de Ingeniería Informática
Albacete, España
alejandro.moya4@alu.uclm.es

Miguel Ángel Sánchez Cifo
Máster Universitario en Ingeniería Informática
Escuela Superior de Ingeniería Informática
Albacete, España
miguelangel.sanchez15@alu.uclm.es

Jorge Valero Molina
Máster Universitario en Ingeniería Informática
Escuela Superior de Ingeniería Informática
Albacete, España
jorge.valero1@alu.uclm.es

Alejandro Zornoza Martínez
Máster Universitario en Ingeniería Informática
Escuela Superior de Ingeniería Informática
Albacete, España
alejandro.zornoza1@alu.uclm.es

Resumen

En este documento se presenta el diseño e implementación de una aplicación en la nube que es capaz de controlar el acceso a un edificio inteligente, mediante el reconocimiento y control de las caras de las personas. El objetivo principal de este trabajo es mejorar los conocimientos de aplicaciones desplegadas en entornos cloud, mediante la elaboración de un proyecto real.

Palabras clave

Amazon Web Servicies; Rekognition; Reconocimiento de caras; Máquinas virtuales, Cloud, Seguridad, Domótica.

1. Introducción

El objetivo principal del proyecto es la creación de una aplicación que permita reconocer y gestionar rostros de personas, con el fin de llevar a cabo un control de acceso a un determinado lugar.

Para ello es necesario diseñar una arquitectura que gestione las imágenes captadas por una cámara en tiempo real.

El diseño de la arquitectura se basa en la utilización de los servicios que proporciona Amazon Web Services (AWS en adelante), dado que permite cubrir toda la casuística pertinente. Además, dado que tenemos la restricción de realizar el proyecto empleando máquinas virtuales (VM en adelante), se utiliza EC2 de AWS por estar basado en instancias de VMs.

2. Motivación

Con este proyecto se persigue poder llevar a cabo el control de entrada y salida de un acceso a un edificio y/o sala. En la actualidad hay diferentes sistemas para poder llevar a cabo este control, ya sea mediante el uso de sistemas simples como el uso de una tarjeta identificativa o sistemas más sofisticados como el uso de biometría ocular.

Este trabajo se centra en la creación de un sistema de control de accesos de manera sencilla y de bajo coste. Para ello, se pretende realizar el control mediante el uso de un video obtenido en tiempo real a través de una cámara conectada a internet. Para la implementación del sistema se usa la infraestructura y los servicios proporcionados por AWS. AWS es adecuado para nuestra implementación debido a su alto grado de maduración en ciertos módulos imprescindibles para el proyecto, como es el módulo de reconocimiento facial que posee AWS, además, AWS permite acceder a todos sus módulos a un nulo/bajo coste, permitiéndonos realizar una primera aproximación del sistema.

3. Visión general de AWS

La plataforma AWS se encarga de proporcionar todos los módulos que se utilizan para desarrollar la solución. Todo se debe implementar sobre una única cuenta sobre la que se cargan los costes de los servicios utilizados.

El primer paso para integrar el equipo es crear los usuarios que son necesarios. De este modo, se evita la compartición de cuentas y es posible controlar los permisos a los que tiene acceso cada uno de ellos. Esto permite limitar los recursos, evitando que personas no autorizadas realicen ciertas acciones. Un buen recurso es crear grupos para asignar los mismos permisos a varias personas a la vez. Con esto es más sencillo gestionar los permisos entre varios usuarios similares.

Definir presupuestos es otra buena medida para controlar los gastos en servicios. Manteniendo una buena política de notificación de gastos se puede contener el coste de prestación de servicios.

Por otro lado, las políticas son una herramienta muy potente de AWS para asegurarse de que cierto servicio solo tiene acceso a otro recurso definido, es decir, su seguridad es muy alta al definir de manera muy precisa la comunicación entre recursos. Hacer un uso

adecuado de las políticas es de alta prioridad para evitar, por ejemplo, corromper información en un cubo de S3.

A su vez, los roles creados para cada servicio permiten asignar varias políticas a un único servicio o recurso. Un ejemplo es la función Lambda, la cual tiene asignado un rol que, a su vez, contiene varias políticas (administradas por AWS y creadas especialmente para la ocasión).

El servicio Virtual Private Cloud (VPC) permite definir redes virtuales internas entre los servicios de AWS. Para que exista una conexión entre los recursos deben encontrarse dentro de una misma VPC. Incluso la conexión con el exterior debe ser definida explícitamente dentro del servicio cuando se usan VPCs. Es bastante común que existan problemas a la hora de conectar todos los recursos utilizados, pues se utilizan conceptos de redes de computadores tal y como los ha definido AWS, haciendo que la configuración de una subred válida para cierto servicio no lo sea para otro.

La seguridad ofrecida a los clientes está presente en todo momento. Las contraseñas deben cambiarse nada más crear un usuario y requieren un cambio cada cierto tiempo. Sobre las contraseñas se pueden aplicar unos requisitos para asegurar que son fuertes. Además AWS posee un rastreador que inspecciona diferentes páginas web susceptibles de contener claves (como GitHub). Con este sistema se puede alertar al cliente de que sus claves han sido compartidas en público, a la vez que se bloquea el acceso a la cuenta comprometida. De hecho este es uno de los puntos clave por el que se debe evitar cualquier tipo de filtración para que nadie pueda acceder a los servicios.

La plataforma ofrece una gran cantidad de servicios que permiten implementar una solución completa sin recurrir a otros elementos externos. Los servicios utilizados relevantes para este proyecto se describen a continuación.

4. Componentes

A continuación se mencionan los diferentes componentes de AWS de los cuales se han hecho uso para implementar el sistema:

- Amazon Cognito. Encargado de la gestión de identidades y autenticación en la nube de forma simple y segura.
- Amazon EC2. Proporciona una amplia selección de tipos de instancias optimizados para adaptarse a diferentes casos de uso. En este componente se pueden crear diversas VMs.
- Amazon S3. Sirve para crear cubos (bucket) en los que se puede almacenar objetos.
- Amazon Rekognition: Encargado de procesar un *streaming* de video o un conjunto de imágenes. Una de las funcionalidades principales es la detección de rostros.
- Amazon Kinesis. Su función principal es el procesamiento de flujos de información y de su almacenamiento. Estos flujos pueden ser de video (Kinesis Video Stream) o de datos (Kinesis Data Stream).
- Amazon Lambda. Servicio que permite la ejecución de código en respuesta a un evento o conjunto de eventos.
- Amazon DynamoDB. Servicio que ofrece una base de datos NoSQL rápido y flexible.
- Amazon RDS. Módulo que permite la creación de una base de datos relacional en la nube. Las bases de datos que se pueden crear son: Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database y SQL Server.

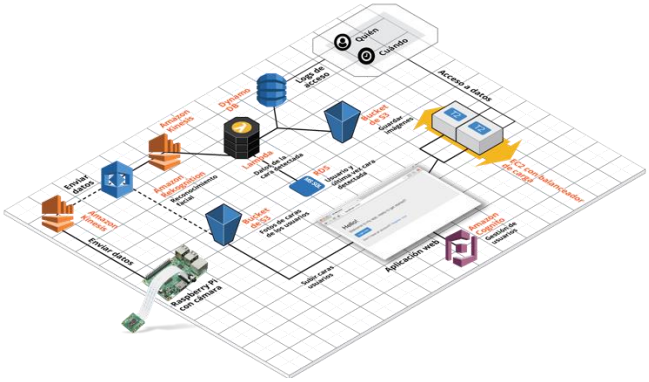

Figura 1. Arquitectura de la aplicación desarrollada

En la Figura 1 se puede apreciar como los diferentes elementos descritos se interconectan, para dar forma y funcionalidad a la aplicación que se describe en este trabajo. La arquitectura ha sido elaborada haciendo uso de la herramienta Cloudcraft [3], la cual no solo permite diagramar arquitecturas desplegables en AWS, sino que también tiene funcionalidades interesantes, entre las que se encuentran el cálculo de costes relacionados con nuestros diagramas o la sincronización con arquitecturas ya desplegadas en la nube de Amazon.

4.1 Amazon Kinesis

El módulo Kinesis de Amazon tiene por objetivo recibir flujos de información, ya sea de video o de datos. El sistema implementado no tiene sentido sin un flujo de video donde poder detectar las caras que pasen por delante de la cámara o cámaras.

En esta primera versión del sistema se ha optado por el uso de una Raspberry Pi 3 Model B. Raspberry Pi es una placa computadora (SBC) muy utilizada en Internet de las Cosas (IoT) por su tamaño reducido y su bajo coste. Tiene una buena relación calidad precio, rondando los 40 € y poseyendo las siguientes características: (1) Chipset Broadcom a 1.2 GHz con procesador ARM Cortex-A 53 de 64 bits y cuatro núcleos, (2) memoria de 1 GB LPDDR2, (3) Bluetooth v4.1 y (4) Wifi 802.11 b/g/n. Además, Raspberry permite la inclusión de módulos que permitan ampliar su funcionalidad. Uno de esos módulos es la "Raspberry Pi Camera V2" que permite incluir una cámara. Todo lo descrito lo hace ideal para nuestro sistema.

El módulo de Kinesis, tal y como está planteado en el sistema desarrollado, se puede dividir en dos partes: Kinesis Video Stream y Kinesis Data Stream. En la Figura 2 se puede ver cómo se mueve el flujo de información, desde que la imagen es captada por la Raspberry hasta que llega la información tratada por el módulo Rekognition a la función Lambda, donde reside la lógica el programa.

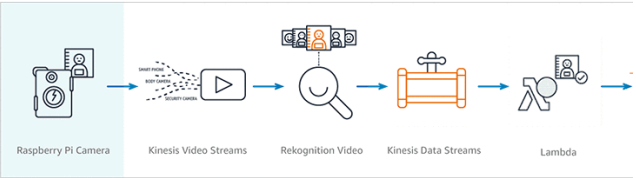

Figura 2. Flujo de datos de Kinesis. Extraído de [2]

4.1.1 Kinesis Video Stream

Este submódulo de Kinesis es el encargado de recibir la información de video del exterior. Para ello es necesario un sistema informático con capacidad de captación de imágenes/videos del exterior y ser capaz de enviar todo lo captado a AWS. Una posible solución de bajo coste es el uso de una Raspberry Pi con su módulo de cámara nativo y la instalación de las herramientas de AWS. Esto es más que suficiente para que se pueda enviar la información a AWS sin muchas complicaciones.

Para que Kinesis Video Stream funcione es necesaria la creación de un *stream* en la plataforma de Amazon donde se aloja el video recibido. Para su creación se tiene que especificar el nombre del *stream*, en este caso es "*PiStream*". Además se tiene que especificar cuanto tiempo será almacenado el video en AWS. En esta implementación se ha optado por la retención durante una hora. La retención de datos es necesaria para que la función lambda, a la hora de detectar una cara, almacene el fotograma donde el usuario ha sido detectado. Hay que tener en cuenta que la retención de video supone un coste extra en el sistema.

Una vez creado el *Stream* de Kinesis Video Stream se procede a la ejecución del programa encargado de enviar las imágenes al *Stream* creado anteriormente. En nuestro caso dicho programa es ejecutado en una Raspberry Pi. El programa utilizado es el propuesto por Amazon para Raspberry, un ejemplo sencillo que sirve de base para la creación de software más complejo.

Para la ejecución del programa se debe especificar los siguientes parámetros: (1) el nombre del *Streamer* de video, en nuestro caso "PiStream", (2) el ancho del video, (3) el alto del video del video y (4) velocidad de fotogramas. Con el fin de garantizar el correcto funcionamiento del Streamer, se tiene que especificar el ancho del video a 640, la altura del video a 480 y la velocidad de los fotogramas a 15.

Antes de ejecutar el programa anterior es necesario especificar el usuario de AWS que realiza la conexión de entrada a Amazon, para ello se ha creado el usuario "kinesis-video-raspberry-pi-producer".

Tras la ejecución del programa y accediendo a Kinesis Video Stream desde la consola de AWS, se puede visualizar el video capturado, tal y como se ve en la Figura 3.

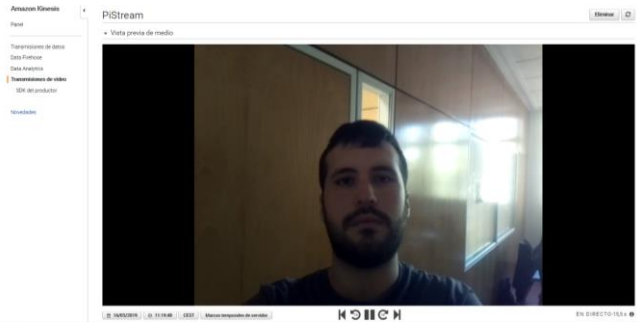

Figura 3. Video capturado por Kinesis Video Stream

4.1.2 Kinesis Data Stream

Este submódulo de Kinesis es el encargado de recibir la información emitida por Rekognition, tal y como se puede visualizar en el apartado de Amazon Rekognition. Para entender la información que recibe Kinesis Data Stream, se recomienda leer en primera instancia el módulo de Amazon Rekognition.

Para empezar a recibir la información enviada por Rekognition, se debe de crear un *Streamer* de datos (no confundir con el *Streamer* de video creado anteriormente). Para crear el Streamer hay que especificar un nombre para su identificación, en nuestro caso "*FacialRecognition*". Además se tiene que indicar la cantidad de fragmentos que puede recibir a la vez. Dado que solo se tiene una cámara funcionando se especifica una capacidad de 1, incrementar este número supone un aumento del coste.

4.2 Amazon Rekognition

El módulo Rekognition de Amazon tiene por objetivo la detección de personas en una imagen o conjunto de imágenes. También permite la detección de personas en un determinado video o un video captado en tiempo real.

En este sistema Amazon Rekognition tiene como objetivo la detección de personas en tiempo real. La ventaja de utilizar Rekognition frente a otras propuestas es que el módulo de detección ya está entrenado y probado por Amazon, garantizando así un funcionamiento óptimo.

El sistema desarrollado plantea el funcionamiento de Rekognition de la siguiente forma: (1) un flujo de video capturado en tiempo real entra en el módulo de Rekognition, (2) se busca en ese flujo de video rostros de personas y las compara con un conjunto de caras previamente almacenadas y (2) se devuelve un flujo de datos con la similaridad de los rostros detectados con los rostros almacenados previamente.

Para almacenar rostros en el sistema es necesario crear una colección de caras en Rekognition. Para cada rostro que se quiere almacenar en el sistema se tiene que especificar: (1) en qué colección de caras se va a almacenar, (2) dónde se encuentra alojado el rostro, en nuestro caso en un Bucket de S3 con nombre "user.pgytic.esiiab.com", y (3) un nombre que sirve de identificador para reconocer qué rostros hay almacenados en la colección de caras. Es importante que por cada persona que se vaya a registrar al sistema se almacenen tres fotos: una de frente y dos ligeramente de perfil (lateral izquierdo y lateral derecho). Para llevar a cabo el registro de personas en el sistema se utiliza una

página web montada en el módulo de EC2, encargado de subir las imágenes de la persona a S3 y de su registro en la colección de caras de Rekognition.

Una vez creada la colección de caras de Rekognition se tiene que proceder a la configuración de Rekognition. Para ello, se tiene que especificar lo siguiente: (1) qué *Streamer* de video se utilizará, en nuestro caso "PiStream", (2) en qué *Streamer* de datos se enviará los resultados, en nuestro caso "FaceRecognition", (3) qué colección de caras se utilizará en el proceso de búsqueda de rostros y (4) el grado de similaridad aceptado para que se diga que un rostro detectado en un determinado fotograma corresponde con un rostro almacenado previamente en la colección de caras, en nuestro caso se ha establecido una similaridad de 85.5 %.

La información que arrojará Rekognition sobre el Kinesis Video Stream es un JSON, tal y como se puede visualizar en la Tabla 1. Lo más importante de todo el JSON es la parte de "MatchedFaces", ya que es una lista de las caras que han pasado el filtro de similaridad de 85.5 %.

Tabla 1. JSON generado por Rekognition. Extraído de [2]


```
{
  "InputInformation": {
    "KinesisVideo": {
      "StreamArn": "arn:aws:kinesisvideo:eu-west-
1:xxxxxxxxxxxxx:stream/my-stream",
      "FragmentNumber":
"91343852333289682796718532614445757584843717598",
      "ServerTimestamp": 1521903783.723,
      "ProducerTimestamp": 1521903783.589,
      "FrameOffsetInSeconds": 2
    }
  },
  "StreamProcessorInformation": {
    "Status": "RUNNING"
  },
.
      "MatchedFaces": [
        {
          "Similarity": 88.863960,
          "Face": {
            "BoundingBox": {
              "Height": 0.557692,
              "Width": 0.749838,
              "Left": 0.103426,
              "Top": 0.206731
            },
            "FaceId": "ed1b560f-d6af-5158-989a-ff586c931545",
            "Confidence": 99.999201,
            "ImageId": "70e09693-2114-57e1-807c-50b6d61fa4dc",
            "ExternalImageId": "nick.jpeg"
          }
        }
      ]
    }
  ]
}
```


4.3 Amazon Lambda

El módulo Lambda de AWS se encarga de identificar al usuario que corresponde con el rostro que ha detectado Rekognition, tratar dicha información en base a la última vez que ha sido detectado, obtener la imagen para almacenarla en S3 y almacenar en el registro de accesos la información necesaria si procede.

La Lambda se ejecuta cuando se publican datos al Kinesis Data Stream con nombre "FaceRecognition". Para ello se le ha asociado un consumidor llamado "lambda-consumer", que se encarga de transmitir a la lambda la información publicada por Rekognition.

Dicha información tiene el formato especificado en la Tabla 2.

Tabla 2. Formato de la información captada por Lambda

Campo	Valor	
kinesis	Campo	Valor
	kinesisSchemaVersion	Versión del esquema de datos de Kinesis
	partitionKey	Clave de la partición
	sequenceNumber	Número de secuencia
	data	Datos transmitidos codificados en Base64
	approximateArrivalTimestamp	Momento aproximado de llegada del mensaje
eventSource	Servicio que origina el evento	
eventVersion	Versión del evento	
eventID	Identificador del fragmento del flujo que contiene al evento	
eventName	Nombre del tipo de evento	
invokeIdentityArn	Rol de quien ha solicitado la información	
awsRegion	Región del consumidor	
eventSourceARN	ARN del servicio que origina el evento (del consumidor)	

La información del campo data, una vez decodificada, tiene el mismo formato que la salida de Rekognition, mostrado en la Tabla 1.

La Lambda, sobre cada registro que le entregue el consumidor, decodifica el campo "data" obteniendo la salida de Rekognition. Los registros que no contengan información en el campo FaceSearchResponse son descartados, mientras que el resto son

pasados a otra función que itera sobre todas las caras detectadas en la imagen (FaceSearchResponse es una lista). Ahí se descartan los registros con caras que no estén en el sistema según Rekognition y se da un trato especial a los que sí lo están:

1. Se suma la similaridad de la cara detectada con todas las fotos de un mismo usuario
2. Se divide la similaridad por 3 (cada usuario tiene 3 posibles fotos)
3. Se devuelve como usuario detectado el que más similaridad tenga, estando por encima de un umbral (85%). Si no se supera dicho umbral, se ignora el registro
4. Se hace una consulta sobre un RDS que tiene como clave primaria los usuarios con acceso al edificio para obtener el *timestamp* de la última vez que su cara fue detectada
5. Si el *timestamp* no supera un umbral (20 segundos) se ignora el registro. Si lo supera se actualiza el *timestamp* en RDS. Si el usuario no tiene registro de último acceso, se crea
6. Sobre los usuarios que hayan pasado el filtro del timestamp, se obtiene del Kinesis Video Stream el fragmento de video en el que la cara fue detectada y se almacena en S3
7. Se almacena en DynamoDB el nombre del usuario, momento del acceso y el nombre del objeto que se ha almacenado en S3

En cada etapa se registra información en CloudWatch detallando el éxito (considerando éxito como seguir lo que se ha indicado arriba, se haya detectado o no cara) o el fracaso (excepción producida que hace que la ejecución salga del cauce indicado) de la operación.

4.4 Amazon DynamoDB

Amazon DynamoDB es una base de datos de pares clave-valor y documentos que ofrece rendimiento en milisegundos de un solo dígito a cualquier escala. Se trata de una base de datos multirregión y multimaestro completamente administrada, con seguridad integrada, copia de seguridad y restauración, y almacenamiento de caché en memoria para aplicaciones a escala de Internet.

Dentro de la nomenclatura de las bases de datos, DynamoDB se encuentra dentro de la categoría de no relacionales, mejorando las prestaciones a la hora de almacenar y acceder a los datos.

En el sistema implementado, la funcionalidad asignada a DynamoDB es la de gestionar datos relacionados con los logs de acceso al edificio, en concreto, la persona y momento concretos. Esta información es posteriormente consumida por la máquina virtual, donde se encuentra la interfaz web y a través de la cual es posible consultar estos datos.

4.5 Amazon RDS

El objetivo principal de Amazon Relational Database Service (Amazon RDS) es configurar, utilizar y escalar una base de datos relacional en la nube. El servicio suministra capacidad rentable y escalable al mismo tiempo que automatiza las arduas tareas administrativas, como el aprovisionamiento de hardware, la configuración de bases de datos, la implementación de parches y la creación de copias de seguridad.

En nuestro escenario concreto se ha decidido usar una base de datos PostgreSQL. La funcionalidad principal de esta base de datos es almacenar con niveles de persistencia adecuados información de los rostros de los usuarios, así como la última vez que fueron detectados por nuestro sistema. Por, tanto, las actualizaciones sobre esta base de datos vendrán provenientes de la correspondiente ejecución de la lambda y de la detección de rostros conocidos.

4.6 Amazon S3

El módulo Lambda de AWS se encarga de almacenar objetos, concretamente imágenes, que son necesarios para el funcionamiento del sistema. S3 permite la definición de cubos o *buckets* en los que se almacenan los objetos. Para el sistema ha sido necesaria la creación de dos cubos:

- users.pgytic.esiiab.com. Cubo donde se almacenan las fotos de los usuarios que se suben a través de la interfaz web. Los usuarios cuyas fotos estén almacenadas aquí serán los que tengan acceso al edificio. Se almacenan dentro de una carpeta con el nombre del usuario
- photologs.pgytic.esiiab.com. Cubo donde se almacenan los fragmentos del stream de vídeo en los que se ha detectado una cara que ha disparado el evento de abrir la puerta

La interfaz web es la encargada de escribir y borrar del cubo "users.pgytic.esiiab.com", mientras que Rekognition es el único módulo que lea dicha información.

El módulo Lambda es el encargado de almacenar los fragmentos del vídeo en el cubo "photologs.pgytic.esiiab.com" y el único módulo con acceso de lectura al cubo es la interfaz web.

4.7 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web que proporciona capacidad informática en la nube segura y de tamaño modificable. Está diseñado para simplificar el uso de la informática en la nube a escala web para los desarrolladores.

Para utilizar Amazon EC2, simplemente se necesita:

- Seleccionar una imagen de máquina de Amazon (AMI) de plantilla preconfigurada para que entre en funcionamiento de inmediato;
- Configurar la seguridad y el acceso a red en la instancia de Amazon EC2;
- Seleccionar los tipos de instancia que se requieren y trabajar con ellas a través de las API de servicio web o la variedad de herramientas de administración proporcionadas;
- Determinar si se desea una ejecución en varias localizaciones, utilizar puntos de enlace de IP estática o adjuntar almacenamiento de bloques continuo a sus instancias;
- El pago se carga solo por los recursos que realmente se consumen, como las horas de uso de instancias o la transferencia de datos.

Las instancias T2 que son las empleadas en este caso (porque cumplen los requisitos del proyecto y son de bajo coste) son instancias de rendimiento ampliable que proporcionan un nivel base de rendimiento de la CPU con la posibilidad de ampliarse por encima del nivel básico. Adicionalmente, en nuestro caso se disponen de dos máquinas virtuales EC2, ya que también se han implementado un balanceador de carga que distribuya la cantidad de peticiones realizadas sobre nuestra aplicación.

El rendimiento base y la capacidad de ampliarse se rigen por los créditos de la CPU. Las instancias T2 reciben créditos de CPU continuamente a un índice fijo en función del tamaño de la instancia, acumulando así créditos de CPU cuando están inactivas y consumiéndolos cuando están activas. Las instancias T2 son una buena opción para una variedad de cargas de trabajo generales, como los microservicios, las aplicaciones interactivas de baja latencia, las bases de datos pequeñas y medianas, los escritorios virtuales, las tareas de desarrollo, los entornos de creación y ensayo, los repositorios de código y los prototipos de productos. Por tanto, poseen el entorno y las características necesarias para satisfacer los requisitos de la práctica.

El módulo de EC2 es utilizado en el sistema para albergar el servidor Apache donde se realizará el despliegue del entorno web de la aplicación, el cual es punto de entrada y salida de la información del sistema, concretamente como entrada, corresponde a la subida de rostros de las personas reconocidas de inicio por el sistema. Al mismo tiempo, funciona de salida de la aplicación a la hora de desplegar el apartado web donde pueden consultarse los logs, es decir, posee el acceso a los datos (de la base relacional) y la monitorización del sistema.

Entre las características de la máquina EC2 creada posee: (1) Procesadores Intel Xeon de alta frecuencia, (2) CPU ampliable, que se rige por créditos de CPU y rendimiento base constante, (3) tipo de instancia de uso general de menor costo e incluida en la capa gratuita y (4) equilibrio de recursos informáticos, de memoria y de red.

Por último, con el fin de garantizar el servicio en todo momento se decidió implementar un balanceador de carga de AWS. El balanceador nos permite redirigir el tráfico en diferentes instancias según la carga de trabajo, con el fin de garantizar que el servicio esté siempre disponible. Tal y como está configurado actualmente el balanceador, hay dos zonas disponibles, la zona eu-west-1a y eu-west-1b. De todas formas, el balanceador de carga no es imprescindible en la implementación que se ha realizado, puesto que la carga que produce la página es bastante baja, toda la necesidad de computo reside principalmente en la Lambda y Rekognition.

4.8 Amazon Cognito

Con la implementación de Cognito se pretende facilitar el acceso de los administradores a la plataforma web de gestión. La idea principal es que solo aquellos que tienen un usuario y contraseña asignados pueden administrar el servicio. Por lo tanto, está bloqueado el registro a cualquier persona externa a la organización que implemente la plataforma.

Aunque existe la posibilidad de implementar un sistema de inicio de sesión personalizado, el servicio prestado por AWS es una opción más inteligente. Cognito posee una seguridad de serie muy elevada, probablemente más que una gestión de credenciales creada desde cero. Además, las opciones de personalización e integración con cualquier software lo hacen una herramienta muy polivalente.

La gran personalización que admite lo hace perfecto para asegurarse de que los administradores usan contraseñas fuertes. Se ha configurado para habilitar el inicio de sesión mediante correo electrónico y nombre de usuario. La contraseña debe tener ocho caracteres al menos y debe incluir números, mayúsculas y minúsculas. Aunque, para facilitar el inicio de sesión durante el desarrollo no se ha activado la autenticación de doble factor, se prevé que sea obligatorio en un entorno de producción.

Todos los usuarios se introducen dentro de un mismo grupo pues, al ser todos administradores, no es necesario dividir los permisos entre ellos. Crear dicho grupo es muy rápido a través de la consola en línea y permite configurar los requisitos de acceso. Después, gracias a las claves API es posible implementarlo en cualquier solución, con cualquier lenguaje soportado. En este caso, al ser PHP el lenguaje escogido, se utiliza el SDK de AWS para Cognito tal y como se detalla más adelante.

4.9 Interfaz web

La interfaz web desarrollada se basa en proporcionar una comunicación interactiva y efectiva con algunos de los componentes utilizados.

La interfaz web se comunica con los módulos de la siguiente manera:

- Amazon Cognito. Se encarga de la gestión de los usuarios que tendrán acceso al sistema. La interfaz web solicita las credenciales del usuario para comprobar sus permisos de acceso al sistema.
- S3 Bucket. Mediante la alta y baja de usuarios, se realizan una serie de operaciones basadas en la subida o eliminación de fotos de perfil de S3 Bucket, respectivamente.
- Rekognition. Mediante la alta y baja de usuarios, se procede al registro o eliminación de fotos de perfil en la colección de caras del módulo Rekognition.
- DynamoDB. Obtención de la información de accesos detectados por el módulo Rekognition, para mostrarla en la interfaz web.

Todas estas acciones han sido desarrolladas usando el SDK de PHP, el cual se basa en una biblioteca de código abierto que facilita la integración de aplicaciones PHP con diferentes servicios AWS.

A través de la comunicación con cada uno de los módulos, se puede vislumbrar el comportamiento de la aplicación web, sin embargo,

vamos a explicar cada una de las posibles acciones a realizar desde la web.

Toda la comunicación realizada se ha basado en la recepción y procesamiento de mensajes JSON específicamente definidos en la documentación de AWS para el SDK de PHP.

El primer contacto con la aplicación se basa en un control de acceso que se encarga de solicitar las credenciales de usuario con el fin de permitir el acceso a la web a únicamente usuarios con permisos desde Amazon Cognitio, Figura 4.

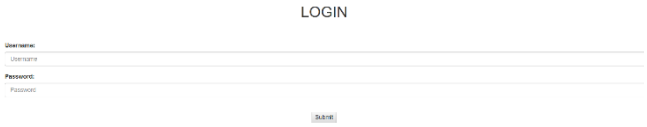

Figura 4 Control de acceso web.

Una vez accedemos a la web, podemos visualizar una tabla con los últimos accesos identificados. En esta tabla podemos visualizar el nombre del usuario detectado y en el momento en el cual ha sido detectado, Figura 5.

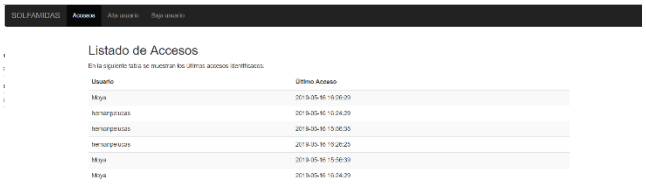

Figura 5. Tabla de accesos identificados.

Esta información se obtiene de la base de datos DynamoDB, donde se van registrando los accesos detectados por el módulo Rekognition.

La interfaz web incluyen dos formularios que nos permiten el alta y baja de usuarios en el sistema.

Para dar de alta un usuario, Figura 6, basta con incluir un nombre en el campo "Nombre" e incluir tres imágenes, las cuales corresponderán con una foto de perfil del usuario, otra del perfil derecho y otra del perfil izquierdo. Se incorpora una limitación de imagen de 3 MB, para no sobrecargar nuestro sistema de almacenamiento con imágenes demasiado pesadas. Además, el módulo Rekognition no requiere de una resolución excesiva de imagen para realizar la detección.

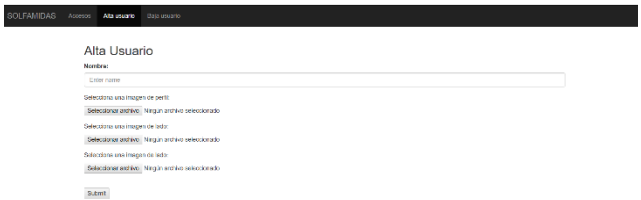

Figura 6. Formulario de alta de usuario.

Una vez hecho esto, se envía el formulario, que se encarga de la subida de dichas fotos a S3 Bucket y registrarlas en la colección de caras del Rekognition.

Por el contrario, se dispone del proceso inverso, que se encarga de dar de baja un usuario, Figura 7. Para eliminar un usuario basta con introducir su nombre de usuario para proceder a eliminar sus fotos del almacenamiento en S3 Bucket, además de su eliminación en la colección de caras del Rekognition.

Figura 7. Formulario de baja de usuario.

5. Métricas de evaluación

Con el fin de demostrar la efectividad de la implementación llevada a cabo, se han realizado una serie de pruebas. Las pruebas realizadas son:

- Prueba de funcionamiento de Rekognition. Se realizaron 10 iteraciones con una duración de 10 segundos donde sobre cada iteración aparecieron de 5 caras registradas en el sistema.
- Prueba de carga sobre EC2. Se utilizó la herramienta Low Orbit Ion Cannon (LOIC). Este programa permite realizar numerosas peticiones (TCP/UDP/HTTP) a una URL o dirección IP.
- Uso normal de la Lambda. Funcionamiento normal con dos usuarios delante de una cámara durante 2 minutos.

5.1 Prueba de funcionamiento de Rekognition

Tras realizar la prueba se ha obtenido como resultado que todas las caras han sido detectadas sobre cada iteración. En algunos casos se detectaron varias caras sobre el mismo fotograma. Esto nos demuestra que Rekognition y la Lambda implementada funciona correctamente.

5.2 Prueba de carga EC2

Se han realizado 6500 peticiones HTTP sobre la dirección del balanceador de carga, para ver cómo se comporta el sistema ante este brusco aumento de solicitudes, observable en la Figura 8.

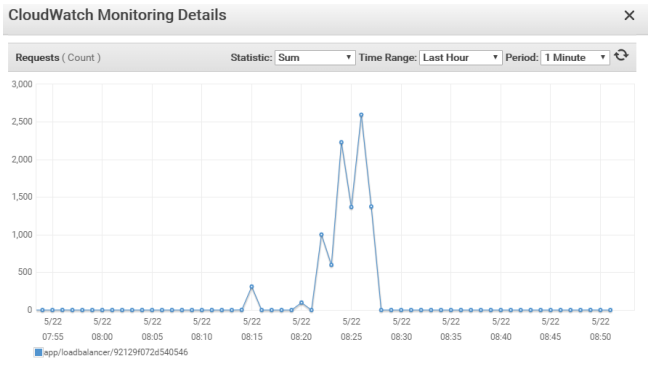

Figura 8. Balanceador de carga - Número de peticiones

5.2.1 Uso de CPU

En cuanto al uso de la CPU, ambas instancias han sufrido un aumento significativo, aunque dentro de valores normales, sin superar el 50% en ninguna de las dos instancias. Además, dicho aumento se ha producido de igual forma en ambas instancias gracias al balanceador de carga, como queda reflejado en la Figura 9 y la Figura 10. Sin embargo, también se observa una ligera diferencia en el uso de CPU de ambas instancias, aunque no muy significativa, teniendo algo más de carga la instancia 2.

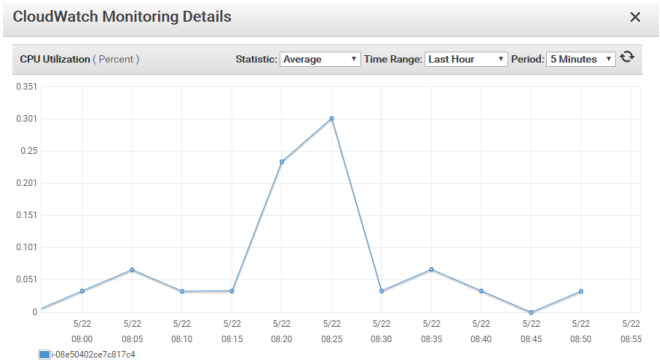

Figura 9. Instancia 1 - Uso de la CPU

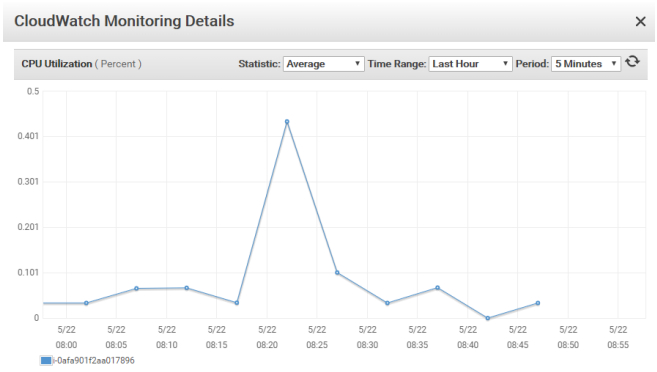

Figura 10. Instancia 2 - Uso de la CPU

5.2.2 Uso de disco

El sistema implementado requiere un uso casi nulo de operaciones de lectura y escritura en disco, por lo que su utilización se mantiene estable y cercana a cero para ambas instancias. La Figura 11 y la Figura 12 muestran la cantidad de bytes leídos de disco en las instancias 1 y dos respectivamente, mientras que la Figura 13 y la Figura 14 muestran los bytes escritos en disco.

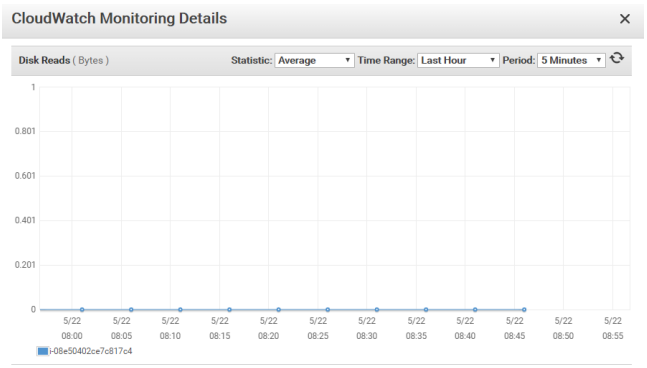

Figura 11. Instancia 1 - Lecturas disco duro (Bytes)

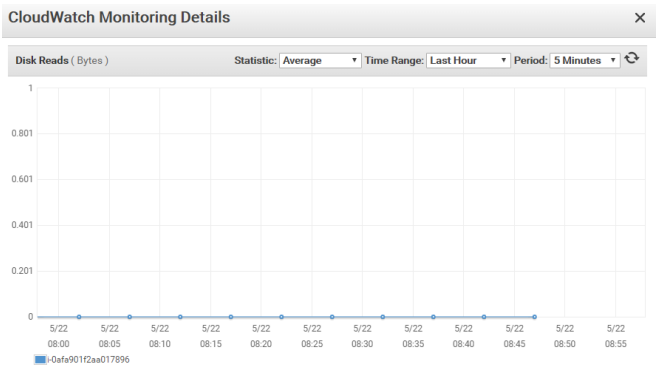

Figura 12. Instancia 2 - Lectura disco duro (Bytes)

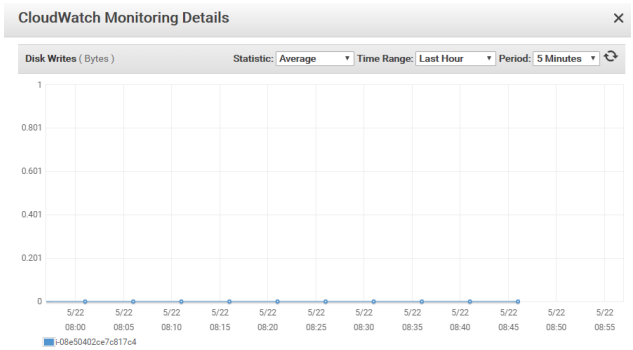

Figura 13. Instancia 1 - Escrituras disco duro (Bytes)

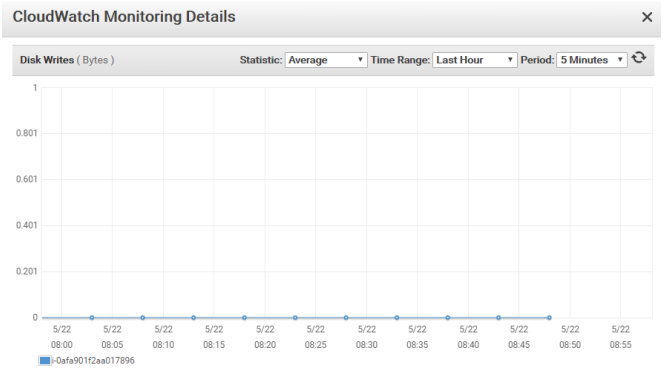

Figura 14. Instancia 2 - Escritura disco duro (Bytes)

5.2.3 Uso de red

La entrada (Figura 15 y Figura 16) y salida (Figura 17 y Figura 18) de bytes de datos en la red aumentan en ambas instancias como consecuencia de semejante número de peticiones en un intervalo corto de tiempo. Se observa que el número de bytes de datos tanto enviados como recibidos por la instancia 2 son mayores, aunque similares. Esto se debe a que el balanceador de carga no ha distribuido las peticiones de forma uniforme tal y como parecía indicar el uso de CPU, sino que la instancia 2 ha recibido una mayor carga de trabajo.

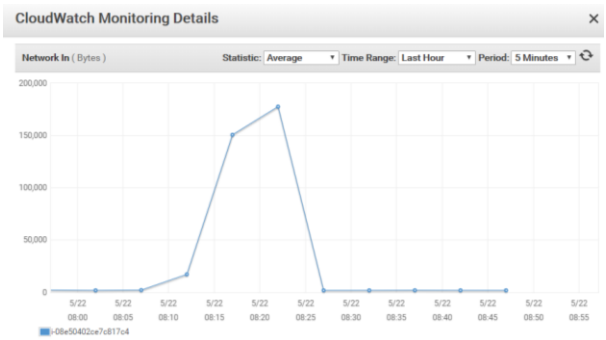

Figura 15. Instancia 1 – Flujo de entrada de Bytes en la red

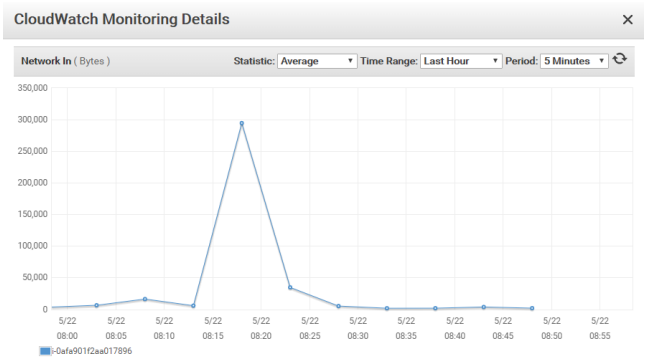

Figura 16. Instancia 2 – Flujo de entrada de Bytes en la red

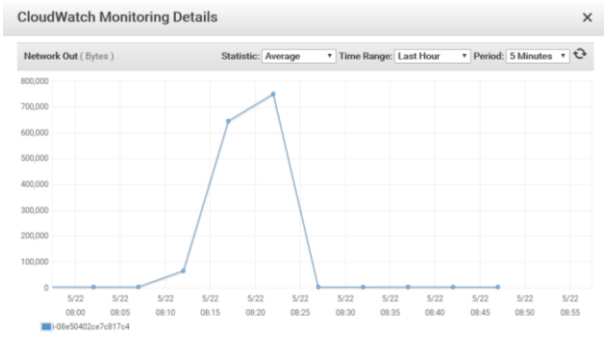

Figura 17. Instancia 1 – Flujo de salida de Bytes en la red

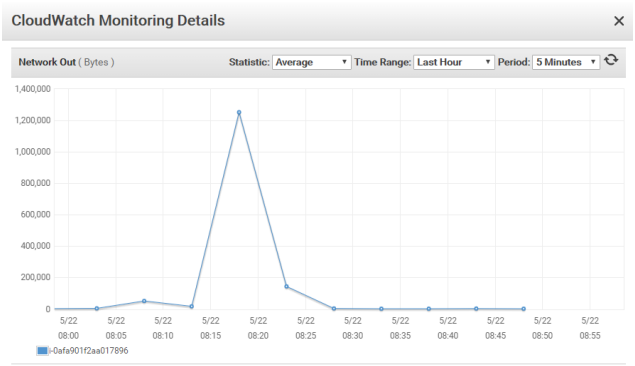

Figura 18. Instancia 2 - Flujo de salida de Bytes en la red

5.3 Uso normal de la Lambda

Durante dos minutos de uso del sistema con dos personas registradas frente a la cámara, se han producido 109 invocaciones de la función Lambda definida, tal y como aparece en la Figura 19. El número de invocaciones no depende de la aparición de usuarios, ya que el módulo Rekognitio publica en el Kinesis Data Stream datos haya detectado o no alguna cara.

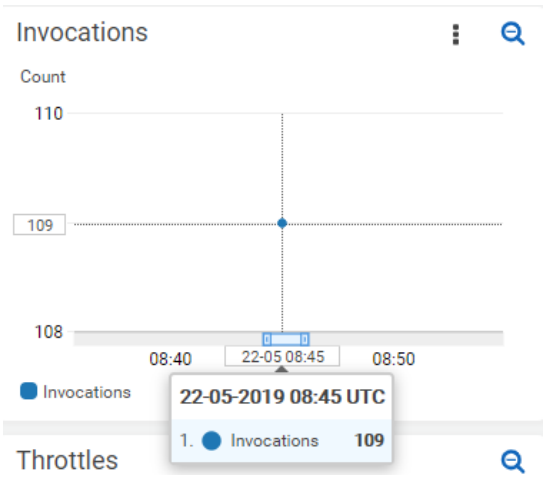

Figura 19. Lamba - Número de invocaciones

Donde el número de rostros detectado tiene un impacto es en la ejecución de la función Lambda, ya que en el caso de no encontrarse ningún rostro en el evento que la dispara la función finaliza, mientras que ante un rostro detectado debe hacer consultas a Amazon RDS y almacenar información en Amazon S3 y Amazon DynamoDB si procede. Esto queda reflejado en la enorme varianza que existe en el tiempo de ejecución observado en la Figura 20, teniendo como media una duración de 31.1 milisegundos y unos tiempos mínimo (no hay nadie en la imagen) y máximo (se han encontrado a los dos usuarios en la misma imagen y hay que crear un registro) de 6’82 y 423 milisegundos respectivamente.

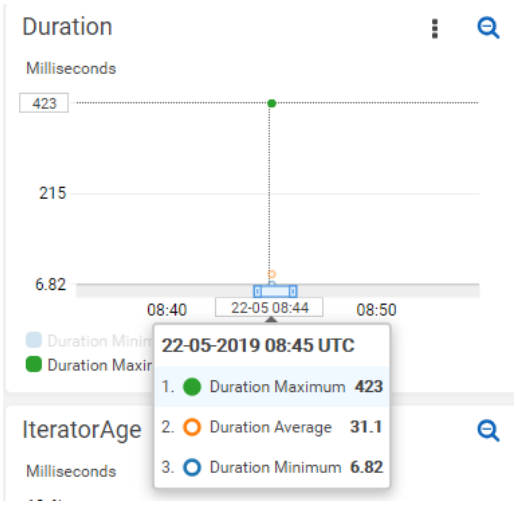

Figura 20. Lambda - Tiempo de ejecución

En cuanto a la tasa de error y acierto de la función Lambda, dado que se ha contemplado en su definición todos los casos de uso y se ha realizado el tratamiento de excepciones pertinente, se tiene según la Figura 21 una tasa de acierto del 100%, sin fallo alguno.

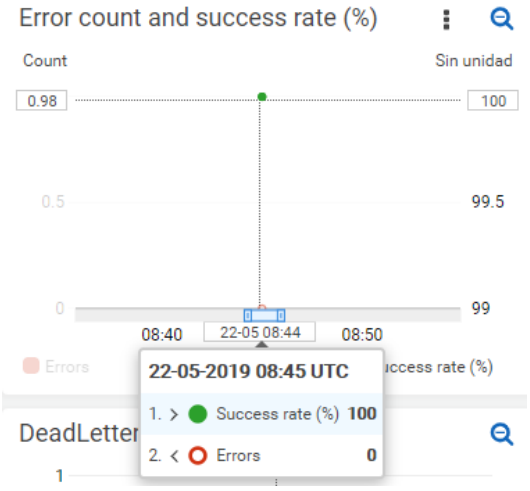

Figura 21. Lambda - Ratio de aciertos y fallos

6. Costes

En relación con los costes, hemos hecho uso de la calculadora de Amazon con el objetivo de realizar una estimación aproximada de lo que costaría el despliegue de nuestra aplicación en la región de Irlanda. A continuación, describiremos dichos costes:

6.1 Amazon Kinesis

Datos recibidos, por GB

Primeros 500 TB/mes	0,031 USD
Siguientes 1,5 PB/mes	0,027 USD
Siguientes 3 PB/mes	0,022
Más de 5 PB/mes	Contacte con nosotros
Conversión de formato de datos, por GB	0,019

6.2 Bucket de Amazon S3

Almacenamiento estándar en S3

Primeros 50 TB/mes	0,023 USD por GB
Siguientes 450 TB/mes	0,022 USD por GB
Más de 500 TB/mes	0,021 USD por GB

Almacenamiento de S3 Estándar – Acceso poco frecuente

Todo el almacenamiento/mes 0,0125 USD por GB

Almacenamiento de S3 Única zona – Acceso poco frecuente

Todo el almacenamiento/mes 0,01 USD por GB

Almacenamiento en S3 Glacier

Todo el almacenamiento/mes 0,004 USD por GB

Almacenamiento S3 Glacier Deep Archive

Todo el almacenamiento/mes 0,00099 USD por GB

S3 Intelligent-Tiering Storage, Frequent Access Tier

Primeros 50 TB/mes	0,023 USD por GB
Siguientes 450 TB/mes	0,022 USD por GB
Más de 500 TB/mes	0,021 USD por GB

S3 Intelligent-Tiering Storage, Infrequent Access Tier

Todo el almacenamiento/mes 0,0125 USD por GB

Almacenamiento S3 Intelligent-Tiering

Monitorización y automatización, todo el almacenamiento/mes 0,0025 USD por cada 1000 objetos

6.3 Amazon Rekognition

Análisis de vídeos

0,10 USD por minuto de video analizado (prorrateado para minutos parciales).

0,12 USD por minuto de video de transmisión en directo analizado (prorrateado para minutos parciales).

Almacenamiento de metadatos de rostros

Precio por 1000 metadatos de rostros almacenados al mes: 0,01 USD. Los cargos de almacenamiento se aplican al mes y se prorratean en el caso de meses parciales.

6.4 Amazon Cognito

Capa de precios (MAU)	Precio por MAU
Primeros 50 000	Gratis
Siguientes 50 000	0,00550 USD
Siguientes 900 000	0,00460 USD
Siguientes 9 000 000	0,00325 USD

Más de 10 000 000 0,00250 USD

6.5 Amazon DynamoDB

Almacenamiento

Los primeros 25 GB almacenados cada mes son gratis

- Los siguientes, 0,283 USD por GB al mes

Solicitudes de lectura y escritura

Tipo de cargo	Precio
Unidades de solicitud de escritura	1,4135 USD por millón de unidades de solicitud de escritura
Unidades de solicitud de lectura	0,283 USD por millón de unidades de solicitud de lectura

6.6 Amazon EC2

CPU virtual ECU Memoria (GiB) Almacenamiento de instancias (GB) Uso de Linux/UNIX

Uso general – Generación actual

t2.micro 1 Variable 1 GiB Solo EBS 0,0126 USD por hora

6.7 Amazon Lambda

Memoria (MB)	Segundos de la capa gratuita al mes	Precio por 100 ms (USD)
128	3 200 000	0,000000208
192	2 133 333	0,000000313
256	1 600 000	0,000000417
320	1 280 000	0,000000521
384	1 066 667	0,000000625
448	914 286	0,000000729
512	800 000	0,000000834
.2 560	160 000	0,000004168
2 624	156 098	0,000004272
2 688	152 381	0,000004376
2 752	148 837	0,000004480
2 816	145 455	0,000004584
2 880	142 222	0,000004688
2 944	139 130	0,000004793
3 008	136 170	0,000004897

6.8 Amazon DB RDS

DB RDS

Precio por hora

Instancias estándar – Generación actual

db.t2.micro 0,02 USD

7. Trabajo futuro

En España se realizan cerca de 2’36 millones de horas extra a la semana. La totalidad de estas horas no son pagadas ni compensadas con horas de descanso. Estas horas extras provoca que los trabajadores trabajen más de las 40 horas obligatorias.

El gobierno, con el fin de paliar este problema, ha escrito un decreto de medidas urgentes de protección social y lucha contra la precariedad laboral en la jornada de trabajo [4]. En este decreto se establece la obligatoriedad de que las empresas lleven a cabo un registro de la jornada laboral efectiva de los trabajadores. Con esta medida se persigue que se paguen las horas extraordinarias y se respeten los descansos. El decreto entra en vigor a partir del 12 de mayo, dejando un margen de adaptación de dos meses. Si no se cumple con lo establecido en el decreto, las empresas se enfrentan a sanciones desde 626 hasta 6.250 euros.

El problema actual es que muchas empresas no saben cómo llevar a cabo este registro, las más pequeñas lo pueden hacer en papel, pero las más grandes necesitan automatizar este servicio.

Para adaptar nuestro sistema a este caso de uso, tan sólo habría que realizar algunos cambios en la estructura de las bases de datos y reconfigurar los módulos para funcionar con varias cámaras.

8. Conclusiones

A través de la utilización de los diferentes componentes que nos proporciona AWS hemos podido obtener una arquitectura escalable y portable que resuelve el problema de gestión de accesos que se nos planteaba.

La utilización de una arquitectura basada en microservicios reduce la complejidad en el desarrollo y mantenimiento de aplicaciones. Sin embargo, la monitorización del rendimiento se vuelve más compleja e impredecible debido a la sincronización con cada uno de los servicios.

AWS nos dispone de multitud de SDKs diferentes que permiten la comunicación con cada uno de los componentes de la arquitectura, algunos de ellos son Java, .NET, Node.js, PHP, Python, C++, Ruby, Android, etc. En este caso, hemos decidido utilizar PHP bajo un servidor Apache, aunque cualquiera de estas opciones es compatible con la arquitectura y se encuentra documentada en la página oficial.

9. Referencias

[1] Amazon Web Services, Servicios de Informática en la nube. https://aws.amazon.com/es/sdk-for-php/

[2] Amazon Web Services. Use facial recognition to deliver high-end consumer experience with Amazon Kinesis Video Stream and Amazon Rekognition Video. https://aws.amazon.com/es/blogs/machine-learning/improve-your-customer-service-using-amazon-kinesis-video-streams-and-amazon-rekognition-video/

[3] Cloudcraft. Visualize your cloud architecture like a pro. https://cloudcraft.co/

[4] Real Decreto-ley 8/2019, de 8 de marzo, de medidas urgentes de protección social y de lucha contra la precariedad laboral en la jornada de trabajo