Grado en Ingeniería Informática

Trabajo de Fin de Grado

Prototipo de sistema de video-seguridad basado en cloud computing y serverless

---

Alejandro Rodríguez Angulo

Tutor/es

D. Carmelo Cuenca Hernández

Dª. Francisca Quintana Domínguez

Las Palmas de Gran Canaria a 19 de noviembre de 2020

Agradecimientos

Me gustaría dedicar este trabajo a mis padres, gracias por el apoyo y la ayuda dada a lo largo de todo mi recorrido por la universidad y por los esfuerzos que habéis realizado para brindarme la oportunidad de formarme.

Me gustaría agradecer a todos los profesores que han contribuido en mi formación durante el Grado en Ingeniería Informática su dedicación y esfuerzo.

Agradecer especialmente a mis tutores, la paciencia, dedicación, tiempo y compromiso dedicados a hacer posible la finalización de este proyecto.

Resumen

Con este proyecto, bautizado como Argus, se pretende desarrollar una aplicación multiplataforma que permita a los usuarios crear un sistema de video seguridad polivalente, escalable, modular y de bajo costo. Con la ayuda de módulos de captura de movimiento, Argus provee a los usuarios de una plataforma de seguridad centralizada para sus oficinas, sus casas o cualquier otra infraestructura que pretenda proteger. El sistema ofrece una aplicación móvil y tantas cámaras como el usuario desee. Toda la infraestructura está implementada en la nube con la ayuda de la plataforma Amazon Web Services, utilizando arquitecturas y patrones serverless. Para el desarrollo de la aplicación móvil se ha utilizado Flutter como framework de desarrollo.

Palabras clave: Computación en la nube, aplicación móvil, videoseguridad, AWS, Flutter, Serverless Framework, Docker, microservicios, Raspberry Pi.

Abstract

This project, named Argus, aimed to develop a multiplatform application that allows users to create a multipurpose, scalable, modular, and low-cost video security system. Using motion capture modules, Argus provides users with a centralized security platform for their offices, homes, or any other infrastructures that they intend to protect. The system does not only offers a mobile application but also provides users with as many cameras as they require for motion detection. All the infrastructure is implemented in the cloud using the Amazon Web Services cloud platform as a cloud provider. The infrastructure design follows the serverless architectures and patterns. The Flutter Framework was the technology selected for developing the mobile application due to its multiplatform characteristics.

Keywords: Cloud computing, mobile application, video-security, AWS, Flutter, Serverless Framework, Docker, microservices, Raspberry Pi.

Índice

1. Introducción

En este apartado se pretende introducir el proyecto al lector. A lo largo del mismo se describe la motivación del proyecto, así como los objetivos planteados.

1.1 Motivación

Durante los últimos años, hemos visto cómo la industria de la videovigilancia se ha infiltrado en nuestros hogares. Los sistemas de videovigilancia han pasado de ser algo propio de empresas y comercios a formar parte de un gran número de viviendas. Estos sistemas han ido evolucionando, abaratando con ello los costes y ofreciendo cada vez más características.

La creciente demanda de sistemas de videoseguridad en los últimos años, ha provocado la aparición de distintas empresas enfocadas al negocio de la videoseguridad, sin embargo parece ser que hoy en día, a pesar de existir un amplio abanico de posibilidades en el mercado, el sector sigue siendo inaccesible, por motivos de precio, para la gran parte de la población.

Este proyecto pretende estudiar una vía de desarrollo de sistemas de videoseguridad accesible para cualquier usuario. Para ello se pretende crear un prototipo que nos permita abaratar costes y escalar en funcionalidades fácilmente, así como proveer al usuario de un sistema de sencilla configuración, permitiendo añadir funcionalidades al sistema sin necesidad de asistencia. Todo ello sin perder calidad en el producto frente a otras opciones en el mercado y posibilitando funcionalidades adicionales como el reconocimiento facial.

En concordancia con lo anteriormente expuesto se plantea el debate de qué tecnologías son las apropiadas para el desarrollo del prototipo. Las opciones son varias, pero dada la naturaleza del problema, las tecnologías elegidas fueron serverless framework para el desarrollo del backend, siguiendo una arquitectura serverless en AWS cloud y flutter para el desarrollo de una aplicación móvil compatible con cualquier dispositivo.

1.2 Objetivos

En esta sección se plantean los distintos objetivos perseguidos tanto a nivel de proyecto como a nivel académico.

1.2.1 Objetivos generales

El desarrollo del prototipo se llevará a cabo teniendo en mente el cumplimiento de los siguientes objetivos en cada paso, se estudiará cada uno de los mismos, se evaluarán las distintas opciones de implementación y se desarrollarán de ser viables.

- Registrar cuando alguien entra/sale en la vivienda/edificio.
- Consultar información acerca de quién entra/sale de la vivienda/edificio.
- Ser capaz de reconocer quién en concreto entra en a la vivienda/edificio.
- Distinguir si entra alguien no registrado.

1.2.2 Objetivos académicos

A continuación se muestran los objetivos académicos perseguidos:

- Ganar experiencia con el desarrollo de aplicaciones en cloud.
- Aprender a desarrollar aplicaciones serverless.
- Aprender a diseñar arquitecturas en el entorno cloud.
- Aprender a desarrollar aplicaciones móviles multiplataforma.
- Planificar y ejecutar un proyecto en su totalidad, pasando por las distintas fases de análisis, diseño y desarrollo.
- Aprender y desplegar una aplicación web en un entorno de producción.
- Aprender a crear dispositivos con raspberry PI.

1.2.3 Justificación de las competencias

Como parte del objetivo de este trabajo se pide elegir una competencia específica, como mínimo, en referencia a la mención cursada. En este caso se cursó la mención de Computación. A continuación se muestran las competencias específicas cubiertas a lo largo del desarrollo de este trabajo.

CP04

Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.

Justificación: Esta competencia se ve cubierta a través de la elaboración de un prototipo en sus distintas fases de análisis, diseño, desarrollo, prueba y despliegue empleando técnicas propias de los sistemas inteligentes, como en este caso es el servicio de reconocimiento facial *Amazon Rekognition*.

CP06

Capacidad para desarrollar y evaluar sistemas interactivos y de presentación de información compleja y su aplicación a la resolución de problemas de diseño de interacción persona computadora.

Justificación: Esta competencia se ve cubierta a través de la elaboración de un prototipo en sus distintas fases de análisis, diseño, desarrollo, prueba y despliegue empleando el framework de desarrollo de aplicaciones *Flutter* para la creación de una aplicación móvil que provee al usuario de un método de interacción con el sistema.

CP07

Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.

Justificación: Esta competencia se ve cubierta a través de la elaboración de un prototipo en sus distintas fases de análisis, diseño, desarrollo, prueba y despliegue empleando técnicas propias de los sistemas inteligentes, como en este caso es el servicio de reconocimiento facial *Amazon Rekognition*.

1.3 Planificación

La planificación inicial está sujeta a los distintos objetivos a cumplir, teniendo en cuenta la dependencia de estos entre sí. Al inicio del proyecto se planteó la planificación mostrada en la tabla 1-1.

Fase	Duración Estimada (horas)	Tareas
Estudio previo / Análisis	40	Tarea 1.1: Estudio del framework serverless Tarea 1.2: Estudio del framework Flutter Tarea 1.3: Estudio de AWS Rekognition
Diseño / Desarrollo / Implementación	205	Tarea 2.1: Desarrollo de la infraestructura backend Tarea 2.2: Desarrollo de la aplicación móvil
Evaluación / Validación / Prueba	20	Tarea 3.1: Prueba de la infraestructura backend Tarea 3.2: Prueba de la aplicación móvil
Documentación / Presentación	15	Tarea 4.1: Elaboración de un readme para github Tarea 4.2: Elaboración de la memoria del TFG Tarea 4.3: Elaboración de una guía de colaboración que permita ampliar la aplicación

Tabla 1-1. Tabla de planificación inicial

1.3.1 Modificaciones

Las distintas tareas se dividieron en fases o iteraciones. Fruto de estas iteraciones se produjeron modificaciones sobre la planificación inicial. La tabla 1-2 muestra la planificación resultado de estas modificaciones.

Fase	Duración Estimada (horas)	Tareas
Estudio previo / Análisis	40	Tarea 1.1: Estudio del framework serverless Tarea 1.2: Estudio del framework Flutter Tarea 1.3: Estudio de AWS Rekognition **Tarea 1.4: Estudio de Docker y docker compose**
Diseño / Desarrollo / Implementación	205	Tarea 2.1: Desarrollo de la infraestructura backend Tarea 2.2: Desarrollo de la aplicación móvil **Tarea 2.3: Desarrollo de productor/cámara**

Fase	Duración Estimada (horas)	Tareas
Evaluación / Validación / Prueba	20	Tarea 3.1: Prueba de la infraestructura backend Tarea 3.2: Prueba de la aplicación móvil **Tarea 3.3: Prueba de productor/cámara**
Documentación / Presentación	15	Tarea 4.1: Elaboración de un readme para github Tarea 4.2: Elaboración de la memoria del TFG Tarea 4.3: Elaboración de una guía de colaboración que permita ampliar la aplicación

Tabla 1-2. Tabla de planificación modificada

1.3.2 Iteraciones

Como se ha indicado en el apartado anterior, el desarrollo se divide en distintas iteraciones. Estas iteraciones se corresponden con la ejecución de una o más tareas. En este punto se enumeran las distintas iteraciones y las tareas que las componen. Algunas de las tareas se encuentran divididas en más de una iteración.

Iteración 1. Puesta a punto de la infraestructura web en AWS

Esta primera iteración se compone de las tareas 1.1, 1.3, 2.1, 3.1 y 4.2. El objetivo de esta tarea es el diseño, creación, desarrollo y prueba de toda la infraestructura web necesaria para dar soporte a la aplicación. El resultado de esta iteración sufrió más adelante cambios.

Iteración 2. Puesta a punto del productor/cámara

Esta iteración se compone de las tareas 1.4, 1.3, 2.3, 3.3 y 4.1. El objetivo de esta tarea es el diseño, creación, desarrollo y prueba de los distintos módulos, hardware y software, que componen los productores/cámaras. En este punto se llevaron a cabo cambios sobre el resultado de la primera iteración, con el fin de crear un servicio controlador para las cámaras/productores. El nuevo servicio añadido proporcionaba al usuario la opción de encender y apagar cámaras desde la aplicación.

Iteración 3. Puesta a punto de la aplicación móvil

Esta iteración se compone de las tareas 1.2, 2.2, 3.2 y 4.1. El objetivo de esta tarea es el diseño, creación, desarrollo y prueba de la aplicación móvil. En este punto se llevaron a cabo cambios sobre el resultado de la primera iteración con el fin de modificar los datos devueltos por ciertos endpoints de distintas APIS.

Iteración 4. Puesta a punto de la documentación

Esta iteración se compone de las tareas 4.2 y 4.3. El objetivo de esta tarea es el diseño, y desarrollo de la memoria.

En la tabla 1-3. se desarrollan las distintas tareas planteadas tras la modificación..

Etapas	Descripción
Estudio framework serverless	Tarea enfocada al aprendizaje de *serverless framework*. El cumplimiento de esta tarea es necesario para el desarrollo de la infraestructura.
Estudio framework Flutter	Tarea enfocada al aprendizaje del framework de desarrollo de aplicaciones *Flutter*. El cumplimiento de esta tarea es necesario para el desarrollo de la aplicación móvil.
Estudio de AWS Rekognition	Tarea enfocada al aprendizaje del servicio *Amazon Rekognition*, el servicio de reconocimiento facial escogido para el sistema. El cumplimiento de esta tarea es necesario para el desarrollo de la infraestructura.
Estudio de Docker y docker compose	Tarea enfocada al estudio de la tecnología Docker y el Framework docker compose. Esta etapa es necesaria para llevar a cabo el Desarrollo del productor/cámara.
Desarrollo de infraestructura	Consiste en el diseño y desarrollo de toda la infraestructura necesaria para dar soporte a la aplicación. Esta infraestructura debe desplegarse en *Amazon Web Services* utilizando *serverless framework.*

Etapas	Descripción
Desarrollo de la aplicación móvil	Consiste en el diseño y desarrollo de la aplicación móvil. Esta aplicación será compatible en ambas plataformas, IOS y android. El framework de desarrollo utilizado será Flutter, el cual utiliza el lenguaje Dart.
Desarrollo de productor/cámara	Consiste en el análisis, diseño y desarrollo del módulo productor o cámara que se encarga de tomar capturas que posteriormente son introducidas y procesadas en el sistema.
Prueba de la infraestructura	Consiste en el testeo de la infraestructura backend obtenida como resultado de la fase de Desarrollo de infraestructura.
Prueba de la aplicación móvil	Consiste en el testeo de la aplicación móvil obtenida como resultado de la fase de Desarrollo de la aplicación móvil.
Prueba de productor/cámara	Consiste en el testeo del software y hardware obtenido como resultado de la fase de Desarrollo de productor/cámara.
Elaboración de un readme para github	Tarea enfocada a la redacción de un Readme para cada uno de los códigos desarrollados que faciliten a cualquier desarrollador entender el código implementado.
Elaboración de la memoria del TFG	Tarea enfocada en el desarrollo de la memoria.

Tabla 1-3. Tabla de tareas

2. Análisis

2.1 Estudio de aplicaciones similares

El sistema, una vez desarrollado y pasada la fase de prototipo competiría con un gran número de aplicaciones creadas para ofrecer un sistema de videoseguridad a los usuarios. Las principales ventajas competitivas que ofrece el sistema desarrollado, frente a los demás en el mercado, son el reconocimiento facial y el bajo coste asociado a la tecnología serverless.

A lo largo de este apartado realizaremos un estudio de dos de las principales aplicaciones del mercado. Para llevar a cabo este análisis nos enfocaremos en tres ámbitos, el modelo de negocio, las funcionalidades e interfaz de usuario.

Las aplicaciones objeto de estudio serán My Verisure y WardenCam.

2.1.1 My Verisure

Securitas Direct es una empresa que ofrece a los usuarios un sistema de seguridad basado en el uso de cámaras y sensores. Este sistema está orientado a la protección de viviendas, oficinas y locales. La empresa además ofrece una aplicación móvil a través de la cual se puede gestionar el sistema.

Dada la naturaleza del prototipo desarrollado nos centraremos en el estudio de la aplicación móvil. A continuación analizamos las vistas de mayor relevancia de cara a nuestro prototipo.

Figura 2-1. My Verisure - Vista de menú [7]

La figura 2-1 es una captura del menú principal. En ella se nos presentan las principales funcionalidades ofrecidas por la aplicación. Las funcionalidades destacadas son:

- Consultar el estado de tu Alarma
- Conectar o desconectar tu Alarma de forma remota
- Realizar fotos a distancia
- Pedir asistencia
- Comprobar quién entra y sale de tu hogar o negocio y a qué hora

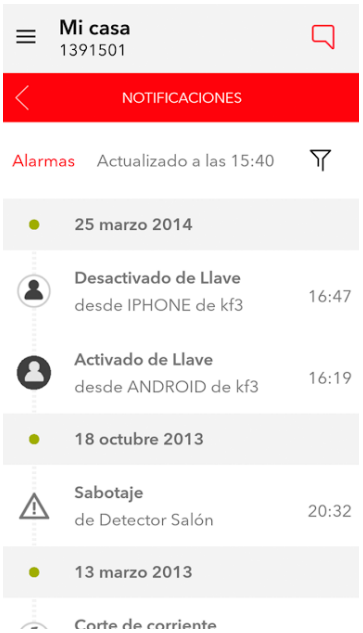

Figura 2-2. My Verisure - Vista de notificaciones [7]

En la figura 2-2 podemos observar el panel de notificaciones que nos ofrece. En este panel podemos encontrar datos acerca del historial de actividad del sistema. Esto incluye información acerca de cuándo se activa y desactiva la alarma y detección de cortes de corriente entre otros.

Figura 2-3. My Verisure - Vista de fotografía [7]

En la figura 2-3 se aprecia la vista de captura. Esta vista muestra una captura tomada a través de la aplicación.

A continuación se muestran las principales características de la aplicación.

	
Tipo	Sistema de videovigilancia
Número de usuarios	Más de 1.000.000 de clientes
Año de lanzamiento	1988
Sitio oficial	https://www.securitasdirect.es/
Dispositivos	Ofrecen un gran número de dispositivos como cámaras, altavoces y sensores de movimiento.
Precio	Variable en función de la vivienda. Se divide en coste de instalación y tasas mensuales. Los costes de este servicio son muy elevados en comparación con el resto planteadas. Los costes mensuales podrían sobrepasan fácilmente los 100€ y puede llegar a costar 300€ o más para un piso unifamiliar. Los costes de instalación varían en función de la propiedad a proteger.

Tabla 2-1. Tabla descriptiva de Securitas Direct [8]

2.1.2 WardenCam

WardenCam es una aplicación de videovigilancia, basada en la utilización de móviles como cámaras. La ventaja de esta estrategia es que no hace falta utilizar un hardware específico. Esta solución ofrece una aplicación móvil a través de la cual se puede gestionar el sistema.

Lo más característico de este software es que, a diferencia de My Verisure, es un sistema de seguridad basado en vídeo. Esto es muy importante ya que el prototipo planteado en este proyecto funciona bajo este mismo principio en su inicio, aunque está preparado para la implementación de otros métodos de detección.

Las principales funcionalidades de este sistema son:

- Visualización en tiempo real

- Escuchar en tiempo real
- Reproducir audios en dispositivos
- Mostrado de alertas ante detección de movimientos

A continuación analizamos las vistas de mayor relevancia de cara a nuestro prototipo.

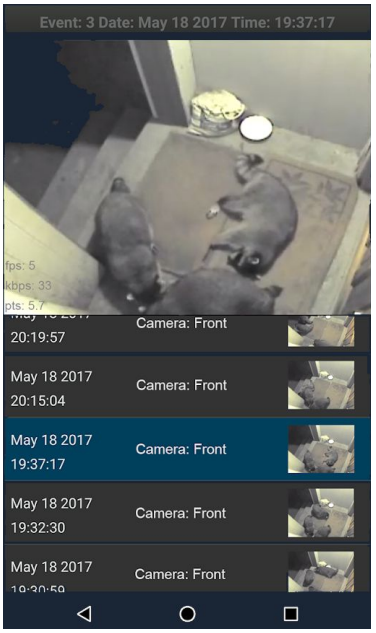

Figura 2-4. WardenCam - Vista de evento [9]

La figura 2-4 se corresponde con la funcionalidad de Mostrado de alertas ante detección de movimientos. En esta captura se aprecia la vista de un evento. Por evento se entiende a un conjunto de imágenes tomadas como producto de una detección de movimiento, en un mismo lugar en un periodo corto de tiempo. Como podemos apreciar esta vista nos muestra un listado de todas las imágenes capturadas a lo largo del evento, así como la imagen del evento seleccionado.

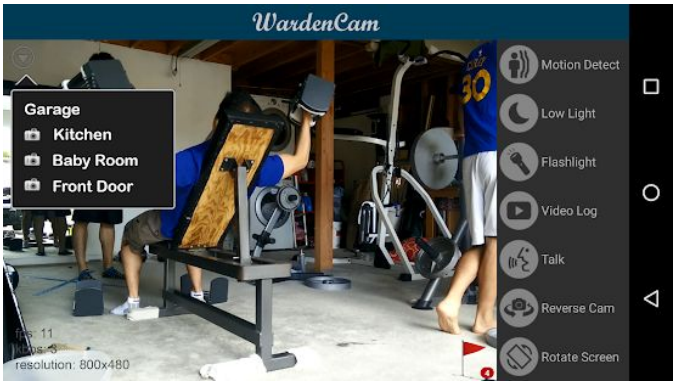

Figura 2-5. WardenCam - Vista de streaming [9]

En la figura 2-5 se aprecia la funcionalidad de visualización en tiempo real. Podemos ver las distintas opciones ofrecidas en el menú de la derecha, entre las cuales se encuentra la funcionalidad de reproducir audios en dispositivos.

WardenCam	
Tipo	Sistema de videovigilancia
Número de usuarios	5.000.000
Año de lanzamiento	2013
Sitio oficial	http://www.wardencam360.com/
Cámara	Se utilizan móviles como cámaras. En su web venden complementos para móviles como trípodes
Precio	$5.99 por mes

Tabla 2-2. Tabla descriptiva de WardenCam [10]

2.1.3 Comparativa

Para finalizar con este apartado se realiza una comparativa entre ambas soluciones analizadas.

Característica	SECURITAS Direct	WardenCam
Sistema	Securitas Direct/ My Verisure	WardenCam
Tipo	Sistema de videovigilancia	Sistema de videovigilancia
Número de usuarios	1.000.000+	5.000.000+
Año de lanzamiento	1988	2013

Característica	SECURITAS Direct	WardenCam
Coste	Elevado variable	Bajo
Encendido y apagado remoto	✓	✗
Hardware específico	✓	✗
Asistencia remota	✓	✗
Multicuenta	✓	✗
Integración cloud	✗	✓
Video en vivo	✗	✓
Reconocimiento facial	✗	✗
Detección de movimientos	✓	✓
Notificaciones ante detección de movimiento	✓	✓
Soporte multicámara	✓	✓
Captura de imágenes	✓	✓
Grabaciones	✓	✓
Visión nocturna	✓	✓
Soporte IOS	✓	✓
Soporte Android	✓	✓

Tabla 2-3. Tabla descriptiva de WardenCam [9] [10]

2.2 Requisitos

El sistema planteado contempla la existencia de un único usuario administrador. Este usuario utiliza la aplicación con el objetivo de crear un sistema de seguridad en su vivienda u oficina, de bajo coste, que le permita obtener información acerca de quién y cuándo accede a su propiedad. Para cumplir con este objetivo se plantean una serie de requisitos de cara al desarrollo del prototipo planteado.

Podemos distinguir dos tipos de requisitos. Los requisitos funcionales y los requisitos no funcionales. En este apartado se exponen los requisitos a exponer siguiendo la clasificación anteriormente expuesta.

2.2.1 Requisitos funcionales

Los requisitos funcionales, son aquellos requisitos necesarios para el cumplimiento de los objetivos del usuario. De acuerdo con esta definición se exponen los distintos resultados en la tabla 2-4.

Id	Descripción
F01	El sistema deberá permitir al usuario consultar el historial de eventos capturados.
F02	El sistema deberá permitir al usuario obtener información del productor asociado a cada uno de los eventos capturados.
F03	El sistema deberá permitir al usuario obtener información de la fecha de los eventos capturados.
F04	El sistema deberá permitir al usuario saber si un evento ha registrado un acceso permitido o si por el contrario ha registrado un acceso no permitido.
F05	El sistema deberá permitir al usuario obtener imágenes de los eventos capturados.
F06	El sistema deberá permitir al usuario obtener información sobre quién ha accedido a vivienda u oficina, en caso de que sea una persona registrada en el sistema.

Id	Descripción
F07	El sistema deberá permitir al usuario registrar nuevas personas en el sistema.
F08	El sistema deberá permitir al usuario asociar un nombre a cada persona registrada en el sistema.
F09	El sistema deberá permitir asociar a una persona varias imágenes de su cara a la hora de hacer el registro.
F10	El sistema deberá permitir al usuario crear sus propios productores.
F11	El sistema deberá permitir al usuario añadir más productores en el sistema.
F12	El sistema deberá permitir al usuario encender los productores de forma remota.
F13	El sistema deberá permitir al usuario apagar los productores de forma remota.

Tabla 2-4. Tabla de requisitos funcionales

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son cualidades de ejecución y evolución que debe tener el sistema, restricciones y condiciones que afectan a los servicios y que ponen límites, tanto al sistema como al proceso de desarrollo. De acuerdo con esta definición se exponen los distintos resultados en la tabla 2-5.

Id	**Descripción**
NF01	El tiempo de aprendizaje del sistema por un usuario deberá ser inferior a 3 horas.
NF02	El sistema debe contar con un manual de usuario que detalle cómo utilizarlo.
NF03	Los productores del sistema deben diseñarse haciendo uso de piezas al alcance de cualquier persona.

Id	Descripción
NF04	Los productores del sistema deben incluir un software de detección de movimiento.
NF05	El software de los productores debe estar desarrollado siguiendo una arquitectura de microservicios basada en contenedores.
NF06	La infraestructura del sistema debe estar desarrollada en Amazon Web Services, utilizando siempre que sea posible servicios de la plataforma.
NF07	El diseño de la infraestructura del sistema debe seguir una arquitectura serverless.
NF08	Debe utilizarse un framework de desarrollo para el diseño, creación y despliegue de la infraestructura.
NF09	Debe utilizarse un servicio con modelo de pago bajo demanda para el reconocimiento facial.
NF10	Debe utilizarse un framework de desarrollo multiplataforma para el desarrollo de la aplicación móvil.

Tabla 2-5. Tabla de requisitos no funcionales

Los requisitos mostrados en la tabla se corresponden principalmente con la usabilidad, tecnologías a utilizar, desarrollo, diseño y eficiencia. Cabe destacar la ausencia de requisitos legales, los cuales se tratan en un apartado más adelante.

2.3 Especificaciones de casos de uso

El sistema planteado contempla la existencia de un único usuario administrador. En este sistema el usuario tiene acceso a una aplicación móvil a través de la cual puede interactuar con este. A continuación, se muestran los diferentes casos de uso que definen las funcionalidades de las que se compone Sentinel.

Caso de uso	Inicio de sesión	
Descripción	La aplicación permite al usuario iniciar sesión.	
Actores	Usuario no registrado.	
Precondiciones	• El usuario tiene una cuenta creada en la aplicación.	
Flujo normal	**Paso**	
	1	El usuario, completa los campos de usuario y contraseña.
	2	El usuario pulsa en el botón de inicio de sesión.
Postcondiciones	• El usuario accede a las funcionalidades de la aplicación.	
Variaciones		
Excepciones		
Observaciones		

Caso de uso	Visualización de los accesos registrados	
Descripción	El sistema permite al usuario obtener una lista de los accesos registrados a la zona protegida.	
Actores	Usuario Registrado.	
Precondiciones	• El usuario registrado debe haber iniciado sesión en la aplicación. • El usuario debe tener instalado como mínimo un productor.	
Flujo normal	**Paso**	**Acción**
	1	El usuario selecciona el icono de la vista de accesos registrados en el menú inferior de navegación.
Postcondiciones	• La aplicación móvil descarga los datos de accesos registrados. • La aplicación muestra la lista de accesos registrados.	
Variaciones		
Excepciones		
Observaciones	• Cada minuto la aplicación móvil descarga los datos de accesos registrados automáticamente.	

Caso de uso	Visualizar imágenes de los accesos registrados	
Descripción	La aplicación permite al usuario obtener imágenes capturadas durante el registro de un acceso.	
Actores	Usuario Registrado.	
Precondiciones	• El usuario registrado debe haber iniciado sesión en la aplicación. • El usuario debe estar en la vista de accesos registrados. • El usuario debe tener instalado como mínimo un productor. • El sistema debe haber capturado como mínimo un evento.	
Flujo normal	**Paso**	**Acción**
	1	El usuario hace click en un evento de la lista.
	2	El usuario hace click en el icono de imágenes.
	3	El usuario desliza el dedo en un sentido u otro para visualizar las distintas imágenes tomadas.
Postcondiciones	• La aplicación móvil descarga las imágenes capturadas durante el registro de un acceso. • La aplicación muestra un carrusel con las imágenes.	
Observaciones	• Las imágenes en la vista de imágenes mostradas se alternan de manera automática por defecto.	

Caso de uso	Registrar nuevas personas en el sistema	
Descripción	La aplicación permite al usuario registrar nuevas personas para que sean reconocidas por el sistema.	
Actores	Usuario Registrado.	
Precondiciones	• El usuario registrado debe haber iniciado sesión en la aplicación.	
Flujo normal	**Paso**	
	1	El usuario selecciona el icono de la vista de registro de personas en el menú inferior de navegación.
	2	El usuario le da al botón de la cámara para añadir la cámara.
	3	El usuario saca tres fotos de la cara de la persona a registrar.
	4	El usuario introduce el nombre de la persona indexada en el campo de texto de la vista.
	5	El usuario pulsa el botón de enviar.
Postcondiciones	• Se registra una nueva persona en el sistema. • El sistema reconoce los accesos asociados a esta persona como accesos permitidos.	
Observaciones	• Es conveniente que las fotos de las caras se saquen desde distintas perspectivas.	

Caso de uso	Encender/apagar los productores	
Descripción	La aplicación permite al usuario apagar un producto.	
Actores	Usuario Registrado.	
Precondiciones	• El usuario registrado debe haber iniciado sesión en la aplicación. • El sistema debe tener algún productor instalado. • El sistema debe tener algún productor encendido.	
Flujo normal	**Paso**	
	1	El usuario selecciona el icono de la vista del controlador en el menú inferior de navegación.
	2	El usuario pulsa el icono de encendido/apagado del productor cuyo estado quiera cambiar.
Postcondiciones	• El estado del productor cambia a apagado. • El sistema deja de analizar imágenes procedentes del productor apagado.	
Variaciones		
Excepciones		
Observaciones		

Caso de uso	Encender los productores	
Descripción	La aplicación permite al usuario encender un productor.	
Actores	Usuario Registrado.	
Precondiciones	• El usuario registrado debe haber iniciado sesión en la aplicación. • El sistema debe tener algún productor instalado. • El sistema debe tener algún productor apagado.	
Flujo normal	**Paso**	
	1	El usuario selecciona el icono de la vista del controlador en el menú inferior de navegación.
	2	El usuario pulsa el icono de encendido/apagado del productor cuyo estado quiera cambiar.
Postcondiciones	• El estado del productor cambia a encendido. • El sistema comienza a analizar imágenes procedentes del productor encendido.	
Variaciones		
Excepciones		
Observaciones		

Caso de uso	Encender los productores	
Descripción	La aplicación permite al usuario añadir productores al sistema	
Actores	Usuario Registrado.	
Precondiciones	• El usuario registrado debe haber iniciado sesión en la aplicación.	
Flujo normal	**Paso**	
	1	El usuario selecciona el icono de la vista del controlador en el menú inferior de navegación.
	2	El usuario pulsa el icono de añadir productor.
	3	El usuario completa los campos de nombre y secreto de productor.
	4	El usuario pulsa el botón de añadir.
Postcondiciones	• El sistema acepta cualquier evento procedente de un productor que tenga el nombre y secreto especificados.	
Variaciones		
Excepciones		
Observaciones	Esto permite al usuario añadir cualquier tipo de productor siempre y cuando pueda ejecutar peticiones http.	

2.4 Mockups y diseño de la interfaz

Tras establecer los requisitos de la aplicación y analizar las alternativas estudiadas en el apartado de estudio de aplicaciones similares, se realizó un mockup para mostrar el diseño de la interfaz de usuario de la aplicación móvil. En él se incluyen todas las vistas planteadas de la aplicación. En el diseño de estas vistas es clave la influencia de los competidores previamente analizados.

A continuación se explican las distintas vistas obtenidas como resultado de esta etapa.

2.4.1 Vista de accesos registrados

Esta vista tiene como objetivo mostrar los distintos accesos registrados por el sistema. Se entiende por acceso, la detección de movimiento por parte de alguna cámara del sistema así como su consecuente análisis. Esta vista muestra una lista de accesos registrados, dando acerca de cada uno información sobre si se ha detectado a alguien registrado en el sistema o no y la fecha del mismo. Se plantea además la opción de obtener más información de los accesos mediante la selección de su correspondiente elemento de la lista. Esta información incluye, por ejemplo, quién ha sido detectado, de ser alguien indexado en el sistema.

Figura 2-6. Mockups - Vista de accesos registrados

2.4.2 Vista de acceso registrados

Esta vista tiene como objetivo mostrar toda la información de un acceso registrado en una única vista. Esto incluye si se ha detectado a alguien registrado en el sistema o no, la fecha del acceso, el nombre de la cara registrada (de ser posible), el productor asociada al acceso registrado y las imágenes capturadas.

Para ello, la vista muestra un cuadro que contiene una de las imágenes capturadas. Estas imágenes se van alternando con el fin de mostrar todas. Además se muestra el resto de los datos requeridos a continuación de la sección de las imágenes.

Figura 2-7. Mockups - Vista de acceso registrados (acceso permitido)

Figura 2-8. Mockups - Vista de acceso registrados (acceso no permitido)

2.4.3 Vista de indexado de caras

Esta vista tiene como objetivo permitir al usuario añadir nuevas caras conocidas a su sistema. De esta forma, cuando esas nuevas caras sean detectadas, en la vista de accesos registrados se mostrarán como tal.

Para ello, esta vista pide al usuario introducir tres selfies de la cara con distintas perspectivas. Una vez realizados los tres selfies, se muestra un campo de texto para introducir el nombre de la persona que está siendo indexada en el sistema. Cuando se completan ambos campos se permite al usuario llevar a cabo el registro pulsando en el botón de enviar.

Figura 2-9. Mockups - Vista de indexado de caras

2.4.4 Vista del controlador

Esta vista tiene como objetivo proveer al usuario de cierto control sobre los distintos productores del sistema. Con esta vista se pretende permitir al usuario apagar y encender productores.

Para cumplir con su fin, esta vista muestra una lista de elementos representando los distintos productores. Cada Elemento muestra el nombre del productor y el estado del mismo (encendido o apagado), además incluye un botón que permite cambiar el estado del productor.

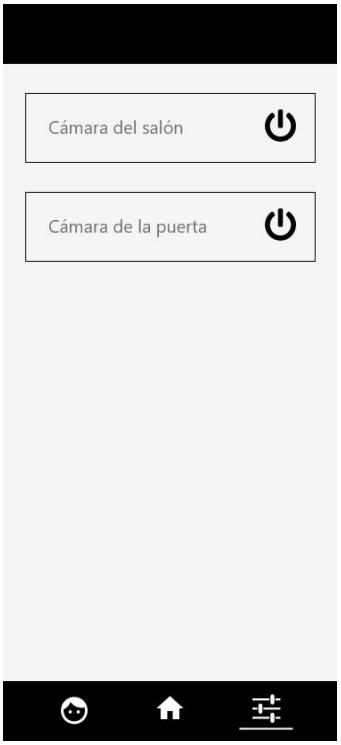

Figura 2-10. Mockups - Vista del controlador

2.5 Modelo de negocio

Dado que el objetivo del sistema planteado no es otro que estudiar alternativas a los sistemas ya existentes, en este apartado nos limitaremos a tratar tres puntos de vital importancia a la hora de crear un modelo de negocio. En este apartado hablaremos de nuestro mercado objetivo, nuestros factores diferenciadores y los costes del sistema.

2.5.1 Mercado objetivo

El usuario objetivo de este sistema, es una persona interesada en tener acceso a herramientas de seguridad para su casa u oficina, a bajo coste. El usuario al que se orienta no es un usuario que demande servicios como asistencia para robos, como ofrecen algunas empresas en el mercado, sino que busca obtener información acerca de lo que pasa en su propiedad. El sistema se enfoca a un usuario que quiere registrar y ser conocedor de los distintos accesos a su vivienda u oficina y que quiere hacerlo a un bajo coste.

2.5.2 Factores diferenciadores

Hay varios factores que diferencian el sistema planteado del resto presentes en el mercado. A continuación se muestran los principales factores diferenciadores, los cuales pretenden ofrecer ventajas competitivas.

- No se requiere de un hardware específico
- Introducción de reconocimiento facial y diccionarios de caras
- Coste bajo de implementación
- Fácil escalabilidad

2.5.3 Costes del Sistema

Los costes de implementación del sistema se dividen en dos secciones. Por un lado los costes asociados al hardware y por otro los costes de la infraestructura. En esta sección abordaremos ambos.

Costes hardware

Estos son los costes de creación de los productores. El diseño de los productores se detalla más adelante en la sección de diseño.

El coste de los productores está asociado a los costes de adquisición de cada uno de los componentes. Dado que uno de los requisitos de este prototipo es la libre elección del hardware, resulta difícil especificar los costes del mismo. Ante esta situación se exponen los costes de creación del hardware del prototipo implementado desglosados.

Componente	Coste
RaspBerry PI 4 (4GB SDRAM)	64.99€
Cámara	25€
Micro SD	20€

Tabla 2-5. Tabla de costes del hardware

Costes de infraestructura

En esta sección se pretenden analizar los costes asociados a la ejecución de nuestra aplicación. Estos costes se ven fuertemente definidos por la utilización de un patrón serverless. El hecho de haber seguido un diseño serverless nos ha obligado a hacer uso de diversos servicios, cada uno de los cuales tiene su propia política de cobro. Para calcular los costes de nuestra aplicación se sigue la siguiente estrategia:

1. Análisis de coste de servicios de AWS
2. Análisis de coste de funcionalidades implementadas

El resultado del Análisis de coste de servicios de AWS se encuentra en el apartado de Análisis de coste de servicios de AWS en la sección de apéndices. En esta sección nos limitaremos a mostrar el Análisis de coste de funcionalidades implementadas. Para ello abordaremos las cuatro funcionalidades principales, haciendo uso del análisis de coste de servicios de AWS.

Coste de indexación de caras

Para hacer posible esta funcionalidad se ha creado el servicio Face Indexer Service, el cual se trata en detalle en la sección de diseño. Este servicio hace uso de Amazon Rekognition, Lambda, API Gateway, DynamoDB y S3.

Los costes asociados a la indexación de una cara se extraen por medio del siguiente proceso:

1. Enviar una foto tomada a S3 a través de una Lambda haciendo uso de API Gateway.
2. Obtener la captura de S3 e indexarla en un diccionario de Amazon Rekognition, haciendo uso de una Lambda y almacenando los datos asociados al usuario en una tabla de dynamoDb.

El resultado del análisis de este proceso nos indica que el coste de indexar una cara es de 0€. Además acumula un coste mensual de 0,0000105€ por el almacenamiento de los datos en Amazon Rekognition y S3.

Coste de captura de evento

Para hacer posible esta funcionalidad se ha creado el servicio Analyzer Service, el cual se trata en detalle en la sección de diseño. Este servicio hace uso de Amazon Rekognition, Lambda, API Gateway, DynamoDB y S3.

Los costes asociados a la indexación de una cara se extraen por medio del siguiente proceso:

1. Enviar una captura tomada por el productor a S3 a través de una Lambda haciendo uso de API Gateway.
2. Obtener la captura de S3 y analizarla con Amazon Rekognition haciendo uso de una Lambda y de una tabla de Dynamo para construir un evento y almacenarlo en S3.

El coste ejecutar este proceso se extrae del siguiente cálculo:

$$nº\ de\ imágenes\ capturadas\ \times\ 0,0008512€$$

Además acumula un coste mensual que se corresponde con la siguiente fórmula:

$$nº\ de\ imágenes\ capturadas\ \times 0,00000002€$$

Hay que tener en cuenta que la cámara captura dos imágenes cada segundo, siempre y cuando detecte movimiento. Por ello el coste variará en función del número de capturas tomadas.

Coste de obtención de eventos

Para hacer posible esta funcionalidad, se ha creado el servicio Event Service, tratado en detalle en la sección de diseño. Este servicio hace uso de Lambda, API Gateway y S3.

Los costes asociados a la indexación de una cara se extraen por medio del siguiente proceso:

1. Recibir una petición desde la aplicación móvil del usuario en API Gateway.
2. Invocar una Lambda que acceda a S3 obtenga el evento y lo devuelva.

Teniendo en cuenta que este proceso no introduce nuevos datos en S3 ni DynamoDB y que no se espera superar la capa gratuita de Lambda, el coste de esta operación se estima en 0€.

Costes de encendido y apagado de productores

Para hacer posible esta funcionalidad se ha creado el servicio Controller Service, el cual se trata en detalle en la sección de diseño. Este servicio hace uso de Lambda, API Gateway, DynamoDB y S3.

Los costes asociados al encendido y apagado de productores se corresponden con el siguiente proceso:

1. Recibir una petición desde la aplicación móvil del usuario en API Gateway.
2. Invocar una Lambda que acceda a DynamoDB y modifique un registro.

El coste de ejecutar una vez este proceso es de 0,00000144 € ¡.

2.6 Normativa y legislación

Los sistemas de videovigilancia son susceptibles de regirse por la Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal (LOPD), siempre que se realice un tratamiento de datos de carácter personal, que en este caso, como resulta obvio es predominantemente el dato de la IMAGEN. Debe recordarse que para quedar bajo el ámbito de esta Ley bastará con la captación del dato, aun no existiendo grabación o almacenamiento. En función del tratamiento realizado la afectación de la ley será mayor o menor [11].

Aunque el sistema desarrollado sea un prototipo, cabe evaluar los aspectos legales del almacenamiento de datos tales como, el nombre, cuenta de correo, y las imágenes capturadas a tener en cuenta en caso de desarrollar una versión comercial.

Los aspectos legales a considerar son:

- Es particularmente importante considerar el hecho del consentimiento del titular de los datos (la persona cuyo dato recabamos), ya que en algunos casos dicho consentimiento puede que tenga que ser expreso, en otros tácito, etc, en función del tipo de tratamiento que se vaya a realizar, pero siempre deberá existir la información [11].
- En el ámbito personal y doméstico no procede aplicar la LOPD.
- Debe de existir una relación de proporcionalidad entre finalidad perseguida y el modo en el que se traten los datos [11].
- Debe de informarse sobre la captación y/o grabación de las imágenes [11].
- No se podrán obtener imágenes de espacios públicos [11].
- Podrán tomarse imágenes parciales y limitadas de vías públicas cuando resulte imprescindible. A este respecto, decir que puede ser exigible usar mecanismos de enmascaramiento de la porción de imagen correspondiente con vía pública [11].

Las principales normativas a tener en cuenta son:

- **LOPD.** La Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal (LOPD), fue una ley orgánica española que tenía por objeto

garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y también los derechos fundamentales de las personas físicas, y especialmente de su honor, intimidad ,privacidad personal y familiar. Fue aprobada por las Cortes Generales el 13 de diciembre de 1999 y derogada con la entrada en vigor, el 6 de diciembre de 2018, de la Ley Orgánica 3/2018 de Protección de Datos Personales y garantía de los derechos digitales, que adapta la legislación española al Reglamento General de Protección de Datos de la Unión Europea [12].

- **Estatuto de los Trabajadores (Para empresas)**. El Estatuto de los Trabajadores recoge las normas fundamentales existentes en el Derecho Laboral en España. Es, por tanto, el texto legal que regula las relaciones laborales. [13].

3. Diseño

En este apartado abordaremos el diseño de la aplicación. Comenzaremos con el diseño genérico del sistema para posteriormente abordar el diseño de cada uno de los componentes de forma individualizada.

De acuerdo con las especificaciones de los apartados anteriores, podemos identificar tres componentes claramente diferenciados. Estos tres componentes interaccionan entre sí para cubrir todos los requisitos planteados. En la figura 3-1 se pretende representar estos componentes así como sus interacciones.

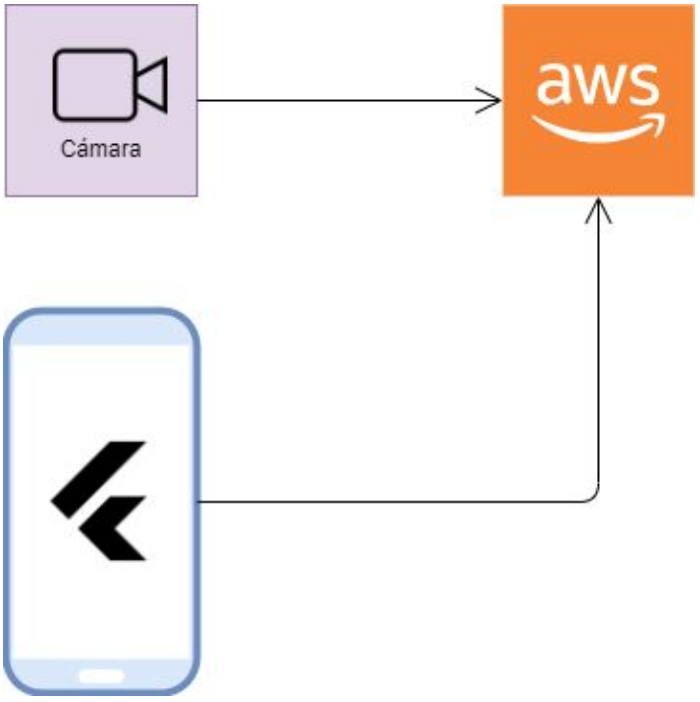

Figura 3-1. Diseño - Interacción entre componentes

En violeta se muestra la cámara, componente encargado de capturar imágenes y subirlas a nuestras bases de datos en Amazon Web Services.

En naranja podemos observar la plataforma cloud de amazon, Amazon Web Servicies. En esta plataforma se aloja toda la infraestructura necesaria para nuestra aplicación.

En azul se representa la aplicación móvil. Con ella podemos acceder a toda la información del sistema.

En la figura 3-2 se pretende representar el flujo de información que atraviesa el sistema y cada uno de los componentes del mismo.

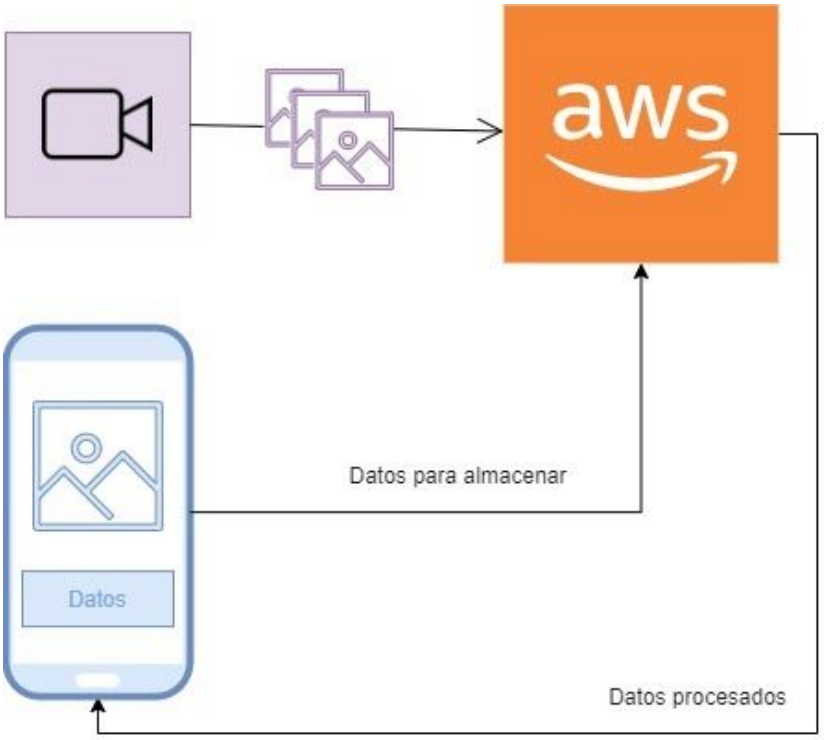

Figura 3-2. Diseño - Flujo de ejecución

De las imágenes anteriores podemos concluir los siguientes puntos:

- La cámara se limita a interaccionar con la infraestructura en AWS proporcionando las imágenes capturadas
- La infraestructura montada en AWS ejecuta transformaciones sobre los datos
- La aplicación móvil sube datos a AWS
- La aplicación móvil consulta los datos procesados de AWS

3.1 Diseño de los productores

En el contexto de este proyecto, un productor es todo elemento que nos permite obtener información del entorno a proteger. Ejemplos de productores podrían ser un micrófono, un sensor de movimiento o una cámara.

En este proyecto se ha diseñado un único tipo de productor dado el tiempo del que se dispone, sin embargo ha sido diseñado para permitir introducir nuevos productores de forma sencilla. El productor diseñado consiste en una cámara con un software de detección de movimiento basado en video. Como parte del desarrollo del proyecto se llevó a cabo el

diseño y posterior creación de un prototipo. Este prototipo pretende ser lo más flexible posible, para posteriormente poder llevar a cabo modificaciones, así como duplicados.

Los criterios seguidos a lo largo del diseño y del desarrollo del prototipo han sido los descritos a continuación:

- ***Diseño modular.*** Se pretende utilizar un diseño modular. De esta forma los distintos componentes Software y Hardware serán independientes entre sí. Esto nos permite iterar de forma mucho más sencilla así como facilitar el mantenimiento, detección de errores, desarrollo y rediseño del prototipo.
- ***Componentes de fácil acceso.*** Se pretende con esto que cualquier persona pudiera crear de forma sencilla un productor. Para ello se tratará de utilizar piezas de bajo coste, accesibles a través de cualquier plataforma de venta online como Amazon o AliExpress.
- ***Instalación sencilla.*** Se pretende facilitar al máximo la instalación del dispositivo, permitiendo así a cualquier usuario poner en marcha su propio sistema. Con el menor esfuerzo posible.

3.1.1 Diseño del hardware

En esta sección se explica el diseño hardware del productor diseñado.

Componentes

En la figura 3-3 se muestran los componentes utilizados para la creación del hardware y su disposición.

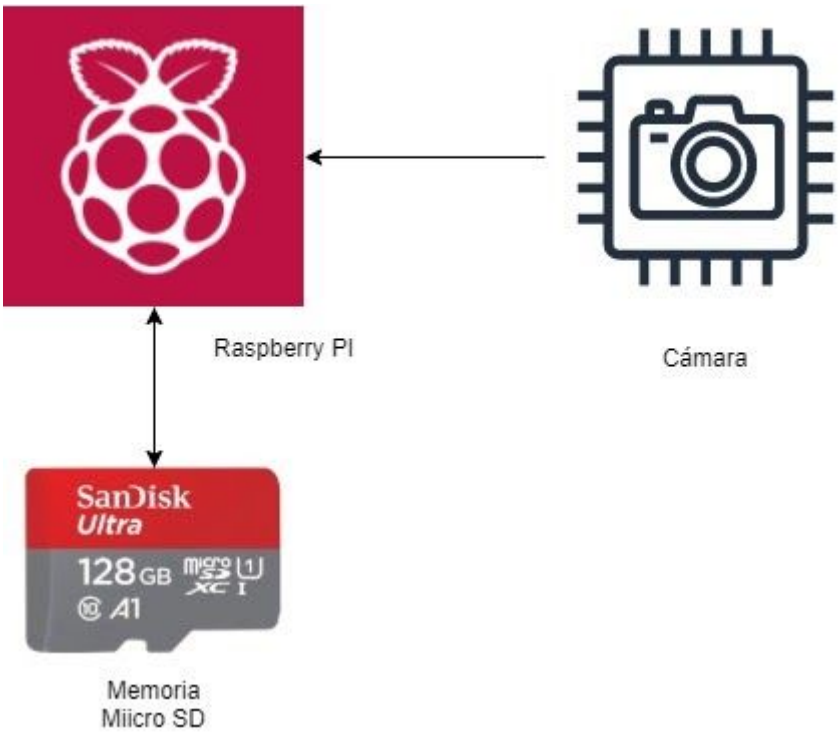

Figura 3-3. Productor - Componentes hardware

RaspBerry PI

Una Raspberry Pi es un ordenador de dimensiones muy reducidas. Este ordenador cuenta con todos los componentes hardware de un ordenador de sobremesa común, o en su defecto algún sustitutivo. Hacer un diseño basado en este tipo de tecnología nos permite ejecutar programas desde el productor. En este caso utilizamos la Raspberry Pi para ejecutar un software de detección de movimiento basado en captura de imágenes.

El modelo de Raspberry Pi utilizado es *Raspberry Pi 4*. Se decidió utilizar la Raspberry Pi 4 y no otro modelo o producto de otra marca debido a la fuerte presencia del dispositivo en el mercado. Este hecho facilita la adquisición del producto contribuyendo así a los objetivos del proyecto.

La ficha técnica del dispositivo puede encontrarse en el apartado 7.1 de la sección de adjuntos.

Cámara

Para la toma de imágenes por parte del productor, es necesario tener un periférico de entrada de tipo cámara. La opción elegida para este componente fue la cámara *Longruner LC26-US*. Se eligió este modelo ya que era ha sido diseñado específicamente para la

Raspberry Pi 4 por lo que ofrece total compatibilidad. Además el precio del dispositivo es bajo en comparación con las otras opciones del mercado.

La ficha técnica del dispositivo puede encontrarse en el punto 7.2 de la sección de adjuntos.

Tarjeta Micro SD

Esta tarjeta Micro SD es utilizada por la Raspberry como elemento de almacenamiento. La capacidad de la memoria utilizada para desarrollar el prototipo es 128 GB.

Otros componentes

Además de los ya descritos es necesario disponer de los siguientes componentes para el desarrollo del productor:

- Cable para la fuente de alimentación (Cable tipo C)
- Cable ethernet (No necesario si se utiliza conexión wifi)
- Bus de conexión entre cámara y Raspberry Pi 4 (Incluido con la cámara)

3.1.2 Diseño del software

El diseño software se llevó a cabo con el objetivo principal de crear una pieza de código capaz de ejecutarse en cualquier plataforma. Con ello pretendemos evitar restricciones a la hora de crear nuevos módulos en distintos sistemas operativos.

Para cumplir con los requerimientos planteados, se decidió utilizar una arquitectura basada en microservicios basada en contenedores. Este tipo de tecnologías nos permite crear entornos virtuales independientes para cada pieza del software. La tecnología escogida para llevar a cabo este diseño fue Docker. Como framework de desarrollo complementario se utiliza Docker compose.

El diseño planteado dio lugar a tres microservicios. Estos microservicios cumplen cada uno con una funcionalidad independiente del resto, y funcionan de forma autónoma. De esta forma facilitamos el manejo, corrección y detección de errores en el software.

La figura 3-4 mostrada a continuación muestra los microservicios diseñados y sus interacciones.

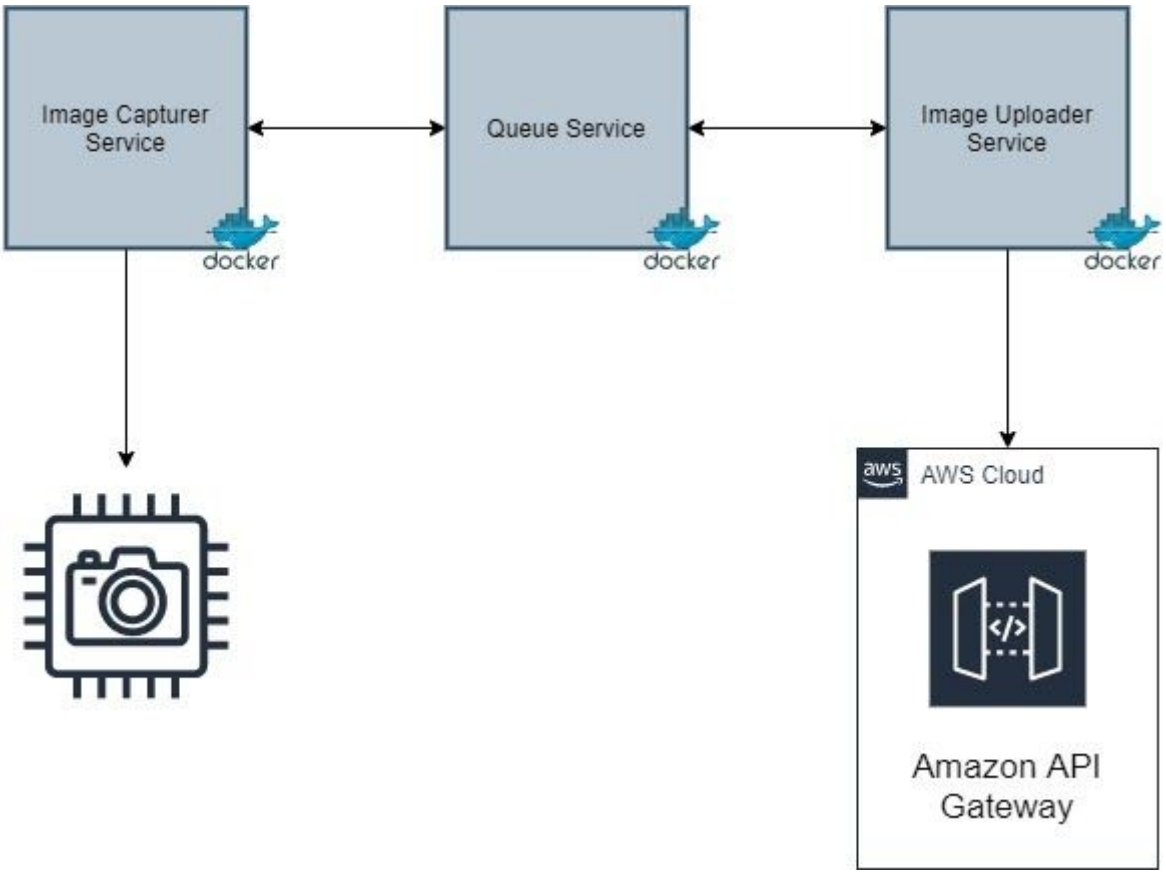

Figura 3-4. Productor - Esquema de microservicios

Image Capturer Service

Este microservicio se encarga de capturar imágenes a través de la cámara, detectar movimientos y guardar en el sistema de ficheros del productor aquellos frames en los que se detectó movimiento.

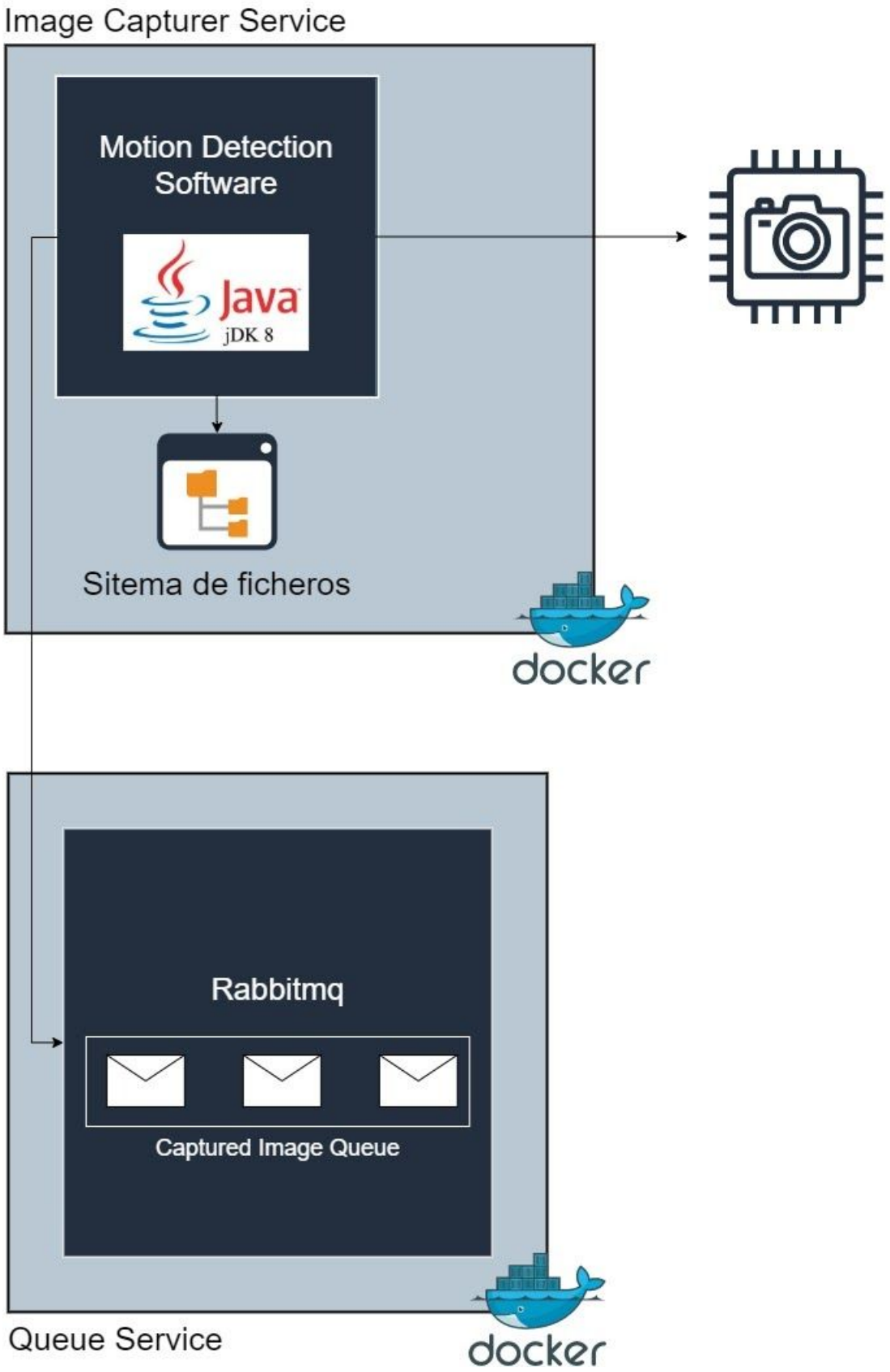

Figura 3-5. Image Capturer Service - Arquitectura

Como podemos observar en la figura 3-5, este servicio utiliza un software de detección de movimiento, creado como parte de este trabajo, basado en imágenes. El software toma imágenes de forma periódica, aplicando a cada una de estas la detección de movimientos. Cuando un movimiento es detectado esta imagen es guardada en el sistema de ficheros, al mismo tiempo el programa notifica al microservicio *Capture Uploader Service* que tiene que subir una imagen a través del microservicio *Queue Service*.

Para permitir al *Image Uploader Service* acceder a las capturas tomadas por el *Image Capturer Service,* se plantea compartir carpetas entre ambos servicios y el equipo donde se ejecutan. De esta forma, cuando almacenamos una captura tomada por el *Image Capturer Service,* estamos guardándola en el equipo y además dando acceso al *Image Uploader Service* a estas, para posteriormente poder subirlas.

El software de detección de movimiento ha sido desarrollado como parte del proyecto utilizando Java 8. Este software compara frames capturados por la cámara y calcula la diferencia pixel a pixel entre estos. Sí la diferencia supera un umbral configurable, se considera que ha habido movimiento y se almacena la captura. Para el tratamiento de imágenes el software utiliza la librería openCV. Esta librería nos simplifica el cálculo de las diferencias entre los frames.

Image Uploader Service

Este micro servicio se encarga de consumir los mensajes de la cola subidos por el servicio de *Image Capturer*, para con esa información acceder a la imagen y subirla a nuestra infraestructura en la nube.

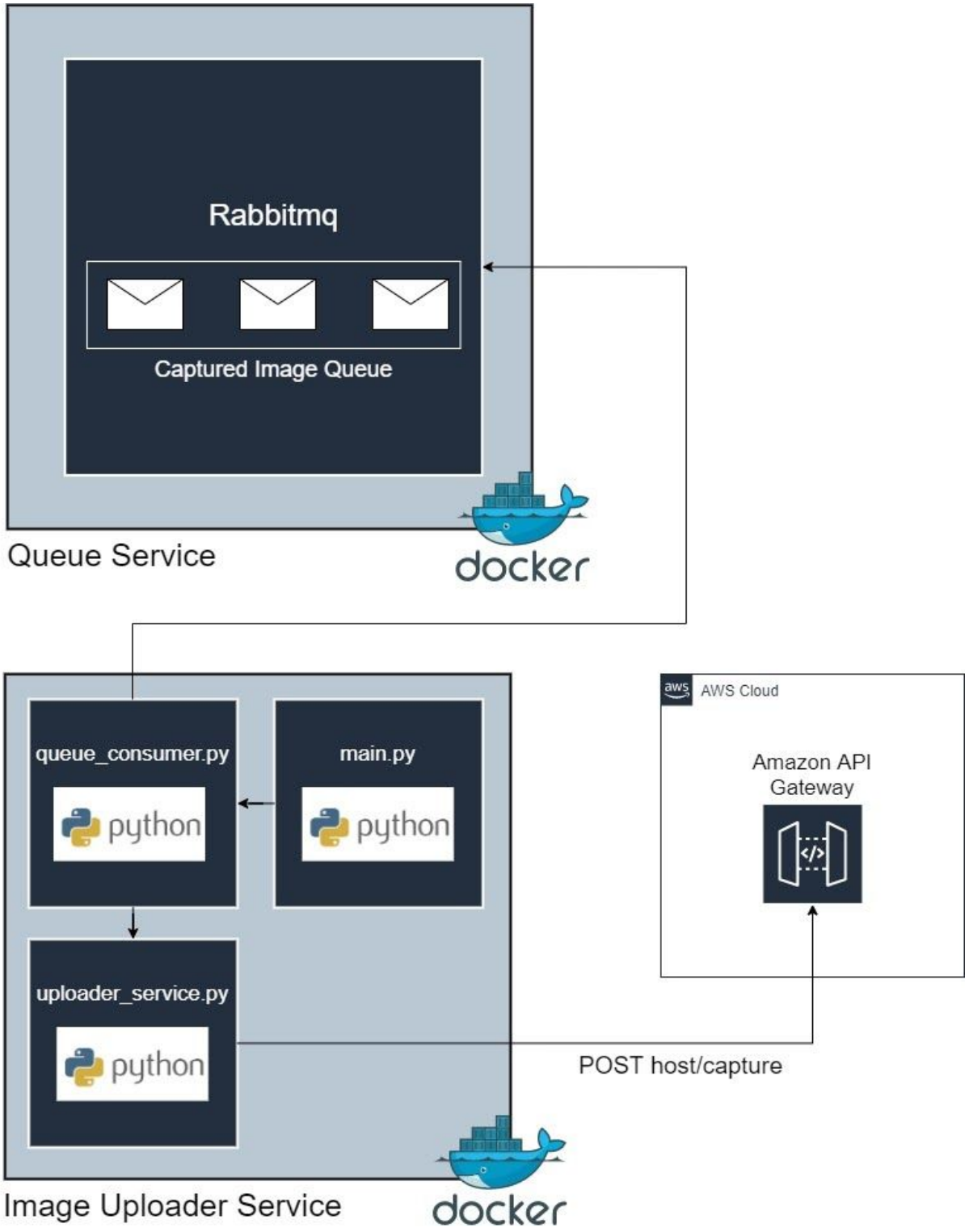

Figura 3-6. Image Uploader - Arquitectura

Cuando este servicio se activa, inicia el consumidor de rabbitmq. Este consumidor se encarga de recoger uno a uno los mensajes dejados por el servicio de captura de imágenes en la cola *captured-image-queue* dentro del servicio *Queue Service*. Este mensaje contiene

la ubicación de la imagen en el sistema de ficheros. Cada vez que se consume una imagen de la cola se invoca a una función de subida de imágenes a AWS.

El software de este servicio está desarrollado en python en su totalidad. Se eligió este lenguaje porque resulta muy sencillo interactuar con rabbitmq utilizando el cliente creado para python.

Queue Service

Este servicio consiste en una cola. Esta cola es una interfaz de comunicación entre el servicio de captura de imágenes y el de subida de imágenes.

Para desarrollar este servicio se ha hecho uso del software de mensajería RabbitMQ. Este software nos permite crear colas. En este caso se ha creado una cola llamada *captured-image-queue* la cual será la utilizada por los servicios para comunicarse.

Para comunicarse con la cola, los distintos servicios necesitan utilizar un cliente de rabbitmq adaptado al lenguaje que utilicen. Los clientes utilizados por cada servicio se muestran en la tabla 3-1.

Servicio	Cliente	Enlace de documentación
Image Capturer	amqp-client	https://www.rabbitmq.com/java-client.html
Image Uploader	pika	https://pika.readthedocs.io/en/stable/

Tabla 3-1. Tabla de clientes de RabbitMQ

3.2 Diseño de la infraestructura

Siguiendo con los objetivos marcados en el comienzo del desarrollo de este trabajo, el diseño de la infraestructura del sistema se caracteriza por seguir un patrón serverless. En línea con este objetivo, el sistema diseñado tiene que, entre otras restricciones, ser desarrollado en la nube a partir de microservicios. Con este fin todo el diseño de la infraestructura está basado en la creación de microservicios en la plataforma Amazon Web Services.

Fruto del diseño de la infraestructura, se proponen cuatro servicios. A pesar de que todos estos servicios están desarrollados en la nube, y aunque acceden a recursos compartidos, son independientes entre sí. Cada microservicio debe, por sí mismo, ser funcional de forma individual y aislada del resto. Estos cuatro servicios se han definido a partir de los principios de serverless. Cada uno de los servicios desarrollados pretende abarcar un conjunto de funcionalidades independientes pero relacionadas entre sí, manteniendo así el máximo sentido semántico. Estos cuatros servicios desarrollados son, Faces Indexer Service, Analyzer Service, Event Service y Controller Service.

Como anteriormente se ha indicado, la infraestructura de esta aplicación ha sido desarrollada utilizando la plataforma en la nube de Amazon, es decir, Amazon Web Services. Esta plataforma nos provee de distintos servicios o funcionalidades, como son bases de datos, servicios de computación o interfaces web, las cuales se utilizan en el diseño de los distintos servicios. Los recursos de Amazon Web services utilizados para el diseño son los expuestos en la tabla 3-2.

Servicio	Descripción
DynamoDB	Utilizado como base de datos. En él se almacenan datos de los distintos usuarios de la aplicación.
S3	Utilizado para almacenar las imágenes subidas tanto por los usuarios a través de la aplicación móvil como por las cámaras en funcionamiento. Además utilizamos este servicio para almacenar la información a mostrar a los usuarios en la aplicación móvil, resultado del análisis de las imágenes capturadas por las distintas cámaras.

Servicio	Descripción
Amazon Rekognition	Utilizado por la app para realizar todas las operaciones relacionadas con el reconocimiento facial. Delegando así las tareas de reconocimiento facial en imágenes capturadas e indexación de caras.
Lambda	Utilizado para ejecutar código en demanda. Cualquier código que sea ejecutado por alguno de los servicios de la infraestructura, estará haciendo uso de este servicio.
Api GateWay	Utilizado para invocar los distintos procesos de los servicios diseñados vía peticiones https desde los demás componentes del sistema.
Amazon Cognito	Utilizado para gestionar los distintos procesos de registro, y login de los usuarios en la aplicación móvil. A través de este servicio podemos securizar las distintas APIs creadas.

Tabla 3-2. Tabla de servicios AWS utilizados en el diseño de la infraestructura

A continuación se muestra una imagen que representa los distintos servicios desarrollados en relación a los recursos de Amazon Web Services utilizados.

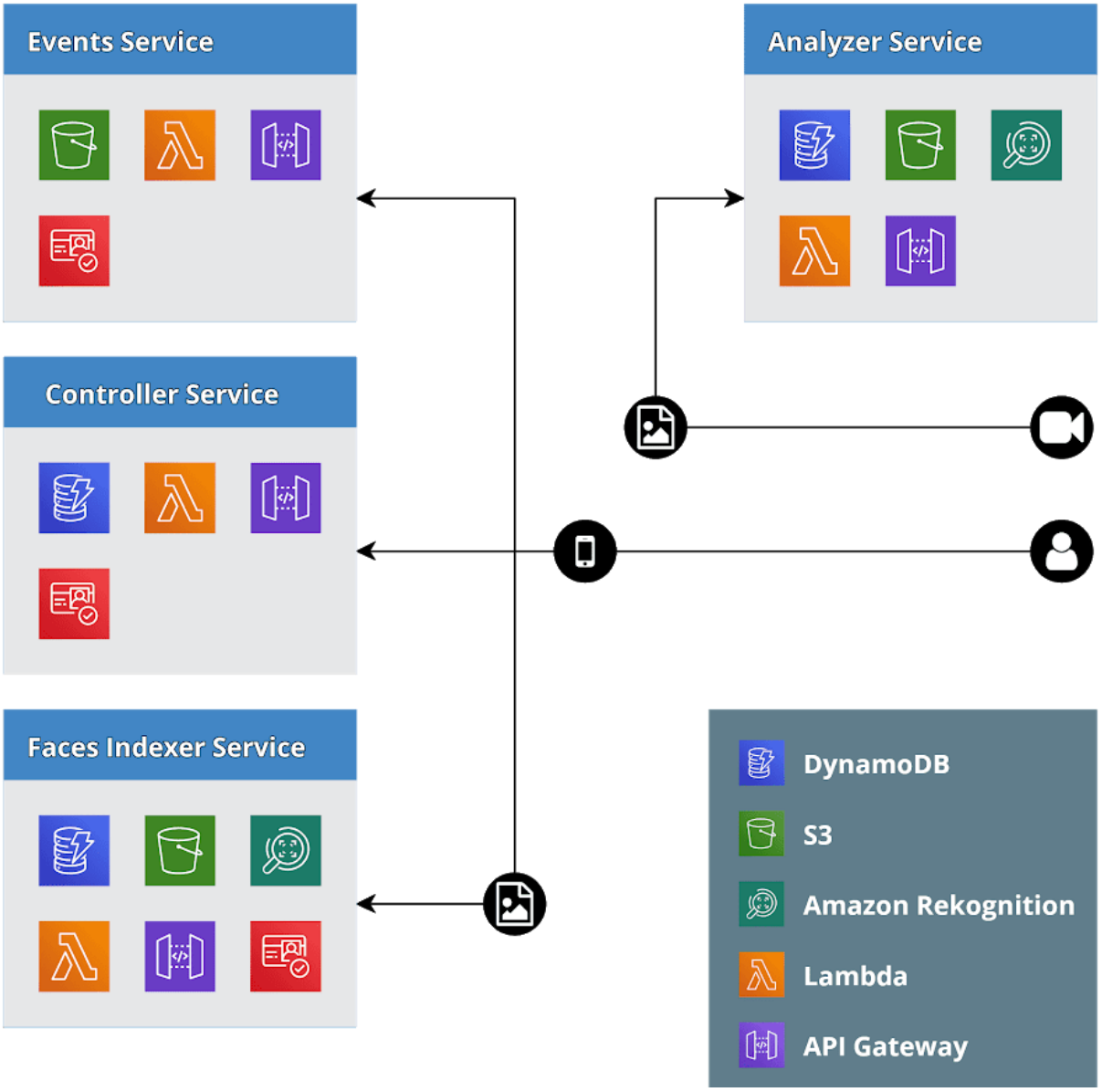

Figura 3-7. Esquema de diseño de Infraestructura

En la figura 3-7 se distinguen los cuatro microservicios a desarrollar conjuntamente con los recursos de Amazon Web Services requeridos para llevar a cabo el desarrollo de cada uno de ellos. Todos los recursos de Amazon Web Services representados se corresponden con los descritos en la tabla 2-7.

3.2.1 Recursos compartidos

Todos los microservicios están conectados entre sí y por ende comparten recursos. A continuación describiremos los recursos compartidos por los distintos microservicios

Tabla de colecciones de caras

Esta tabla hace uso del servicio DynamoDB de Amazon Web Services. En este caso el objetivo de la tabla en cuestión no es otro que persistir las distintas caras registradas por el usuario, a través del servicio FacesIndexer. De esta forma, cuando un una cámara perteneciente a un usuario capture una imagen, el sistema podrá recurrir a los datos almacenados en esta tabla para reconocer a las personas presentes en la imagen y obtener los datos almacenados de la misma. Por lo anteriormente expuesto esta tabla es indispensable tanto para el registro de caras como para el reconocimiento de caras en las imágenes capturadas por el sistema.

Campo / Nombre en tabla	Descripción
Rekonition Face Id / rekognitionFaceID	Se corresponde con el identificador de los datos del proceso de análisis obtenidos por Amazon Rekognintion a la hora de registrar una cara en el sistema. Se utiliza como clave primaria de la tabla, siendo en este caso, el único campo a través del cual se pueden realizar consultas.
Collection Id / collectionId	Se corresponde con el id de la colección de Amazon rekognition a la cual pertenece esta cara. Cada usuario tiene su propia colección y por lo tanto su propio collection id.
Name / name	Se corresponde con el nombre de asociado a la cara registrada en el sistema.

Tabla 3-3. Tabla de colecciones de caras - Tabla de esquema

A continuación se muestra un ejemplo de un registro en la tabla en cuestión.

Rekognition Face Id	Collection Id	Name
46d2d4ac-4cde-4ba4-bcc5-f47c9d1b26b8	foo.example@gmail.com	Alejandro

Tabla 3-4. Tabla de colecciones de caras - Tabla de ejemplo de registros

Tabla de productores

Como se ha expuesto anteriormente en este documento, el sistema permite al usuario tener un número indefinido de cámaras. Estas cámaras son denominadas productores, esto es así porque con ello se pretende permitir la inclusión de nuevos tipos de elementos productores al sistema, estos nuevos tipos podrían ser micrófonos, altavoces, alarmas o sensores de movimiento. Para dar soporte a estos productores hace falta almacenar información que relacione cada uno con su correspondiente cuenta.

La tabla en cuestión nace con el objetivo de permitirnos relacionar las cuentas con sus correspondientes productores. Esta tabla nos permite almacenar el estado de la cámara, es decir, si está encendida o apagada así como los datos de configuración necesarios para el establecimiento de la conexión de la cámara en el sistema.

Campo / Nombre en tabla	Descripción
Account Id / accountId	Se corresponde con el id de la cuenta del usuario. Es la clave primaria de la tabla y por ende es única, es decir, solo hay un registro por cuenta en esta tabla. Este campo se corresponde con el correo empleado a la hora de crearse la cuenta en la aplicación móvil
Producers / producers	Contiene información acerca de los distintos productores asociados a la cuenta en cuestión. Consiste en un conjunto de productores, cada uno de los cuales posee un nombre único, un secreto utilizado para establecer la conexión entre la cuenta y el productor, y por último un estado de encendido o apagado

Tabla 3-5. Tabla de productores - Tabla de esquema de base de datos

A continuación se muestra un ejemplo de un registro en la tabla en cuestión.

Account Id	Productores

alerodriguezangulo@gmail.com	[
	{
	"name": "door-camera",
	"secret": "*********",
	"state": "on"
	},
	{
	"name": "bedroom-camera",
	"secret": "*********",
	"state": "off"
	}
	]

Tabla 3-6. Tabla de productores - Tabla de ejemplo de registros

App S3 Bucket

Este recurso es un Bucket de S3 donde almacenamos todas las imágenes y resultados del procesamiento de datos de los distintos servicios. La estructura de este Bucket se corresponde con el esquema mostrado en la figura 3-8.

Figura 3-8. Almacenamiento S3 - Esquema del bucket

A lo largo del documento haremos referencia a las distintas rutas de nuestro bucket S3 utilizando la correlación mostrada en la tabla 3-7.

Ruta	Seudónimo de la ruta	Tipo de elemento almacenado
/faces	Faces Bucket	Faces o Caras
/captures	Captures Bucket	Captures o Capturas
/events	Events Bucket	Events o Eventos

Tabla 3-7. Almacenamiento S3 - Tabla de seudónimos, rutas y tipos de datos

Faces Bucket

En este bucket o ruta se almacenan todas las caras registradas en las distintas cuentas de la aplicación. Este bucket es utilizado por el servicio *Indexer Service* a la hora de ejecutar la Lambda *Faces Indexer* para subir imágenes de caras registradas por los usuarios en el sistema.

Captures Bucket

En este bucket se almacenan todas las capturas realizadas por los distintos productores de los distintos sistemas montados por los usuarios. Como veremos más adelante este bucket será utilizado por el servicio *Analyzer Service* cuando se ejecute la subida de una captura a través de la Lambda *Captures Uploader*.

Events Bucket

En este bucket se almacenan todos los eventos registrados por los distintos sistemas de los usuarios. Más adelante podremos observar como es el servicio *Analyzer Service,* el que a través de la Lambda *Captures Analyzer,* almacena los eventos creados por la misma en el bucket en cuestión. Por otro lado el servicio *Events Service* hace uso de la información almacenada en este bucket para proveer a la aplicación móvil de la información a mostrar al usuario en referencia a la captura de eventos por parte del sistema. Para acceder a la información de este bucket el servicio *Event Service* hace uso de las lambdas *Find Events* y *Get Event*.

App S3 Modelo de datos

En la 3-8 expuesta a continuación se explica cada uno de los tipos de elementos almacenados en las distintas rutas de nuestro bucket S3.

Tipo de elemento	Descripción
Face o Cara	Archivo jpg que representa una foto de una cara. El nombre de este archivo sigue el patrón *{cuenta}/{nombre de cara}/{tipo de foto}.jpg.* Podemos distinguir tres tipos de fotos, estos tipos son frontal, left-frontal y right-frontal. Además esta imagen contiene metadatos, de estos metadatos podemos obtener la fecha de subida de la imagen y el cliente desde el que se subió.
Capture o Captura	Archivo jpg que representa una captura tomada por un productor. Estos archivos se almacenan de acuerdo con la estructura *{cuenta}/{productor}/{fecha}/{hora}.jpg*. De esta forma tenemos las capturas organizadas por productores y fechas.
Event o Evento	Archivo jpg que representa el resultado del análisis de una captura. Estos archivos se almacenan de acuerdo con la estructura *{cuenta}/{productor}/{fecha}/{hora}.jpg*. De esta forma tenemos los eventos organizados por productores y fechas.

Tabla 3-8. Almacenamiento S3 - Tabla de modelo de datos

3.2.2 Indexer Service

Este microservicio se encarga de indexar las caras de las distintas personas que se registran en el sistema de video-seguridad.

En un inicio se planteó crear un microservicio que reconociera individuos a partir de un video, sin embargo esta opción se descartó por diversos motivos.

- **Coste económico elevado**

  Para analizar el vídeo en tiempo real, sería necesario utilizar un servicio de aws llamado Kinesis VideoStreaming, el cual se encargará de redirigir los datos a la API de Rekognition. El coste de implementar esta arquitectura resulta extremadamente alto en comparación con otras alternativas.

- **Incompatibilidad con el framework serverless**

  Esta solución no sería implementable a través del framework serverless. Esto complica significativamente el proceso de despliegue del servicio. Dado que uno de los principales objetivos es simplificar el proceso de despliegue al máximo, se decidió descartar esta opción.

En contraposición se propone realizar un preprocesado del video para extraer imágenes y enviarlas a la API de Rekognition. Para ello se necesitará crear una API propia, utilizada para subir imágenes capturadas cuando la cámara detectase movimiento.

Este servicio nos provee de dos funcionalidades, implementadas cada una con una función lambda propia. Estas funcionalidades son las representadas en la tabla 3-9.

Funcionalidad	Función lambda
Crear una colección de caras	Collections Creator
Registrar nuevas caras	Faces Indexer y Faces Uploader

Tabla 3-9. Indexer Service - Tabla de funcionalidades

A continuación se muestra el diseño utilizado en la implementación del servicio.

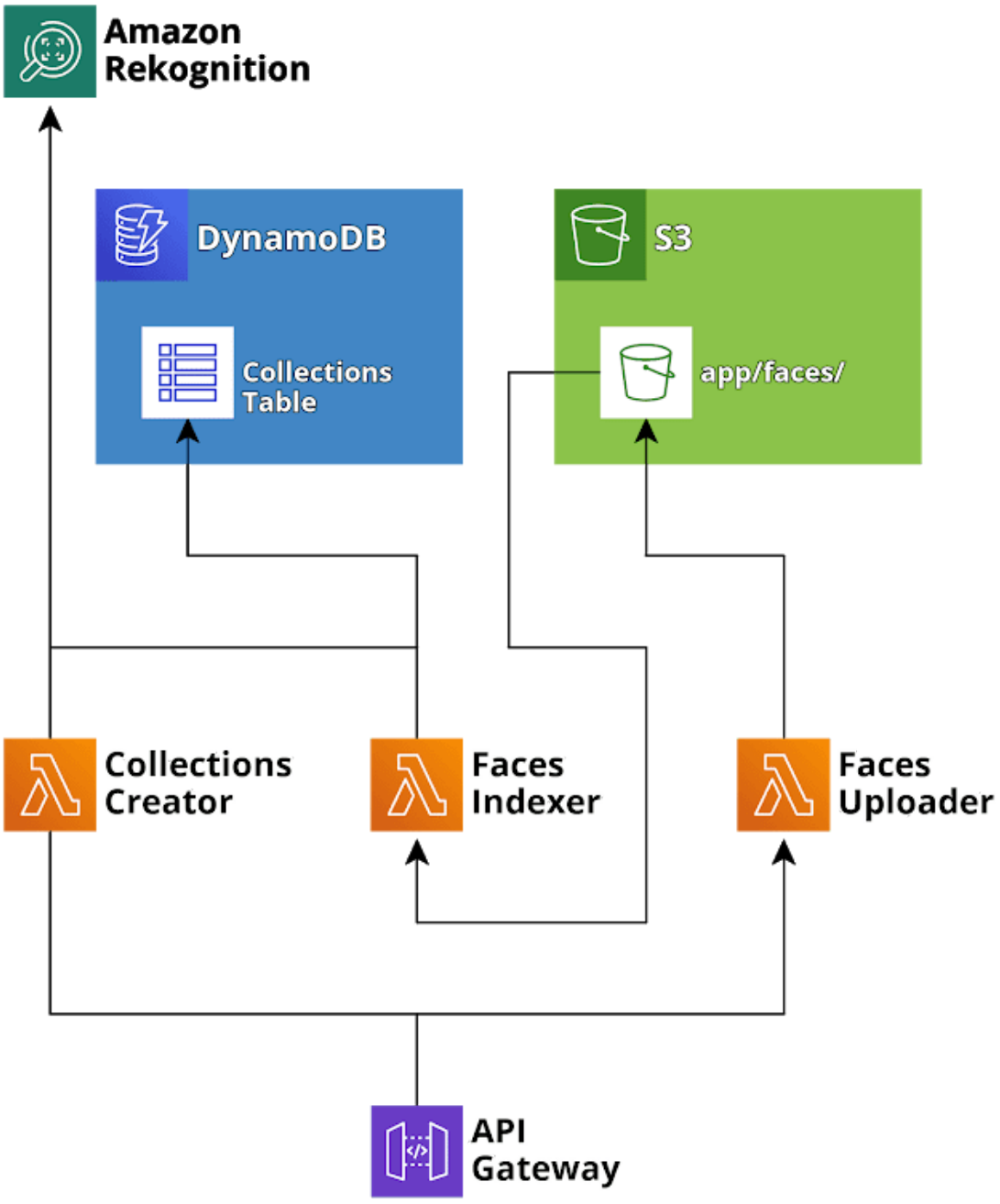

Figura 3-9. Indexer Service - Arquitectura de servicio

Collections creator

Este proceso se ejecuta cuando se crea una cuenta. En este momento se define una colección en amazon web services que se asigna al usuario registrado. La asignación entre la cuenta y la colección se hace a través del nombre de la misma, de esta forma la colección de la cuenta será usuario@gmail.com.

Este componente es una función Lambda. Esta función Lambda tiene como objetivo crear una colección de caras para cada usuario en el servicio Amazon Recognition. Esta función es lanzada por el servicio API Gateway cuando este recibe una petición http de tipo post al endpoint “/collection”. Las distintas colecciones creadas se almacenan en la tabla de colecciones de caras.

Para ejecutar esta lambda es necesario proveerla de los siguientes parámetros de entrada.

Nombre del parámetro	Descripción
Token	Este parámetro es un JWT (**Json Web Token**). Esta ristra aparte de ser necesaria para la autenticación, contiene información acerca de la cuenta del usuario que está realizando la petición. Utilizando esta información somos capaces de acceder al conjunto de eventos asociados al usuario.

Tabla 3-10. Lambda Collection Creator - Tabla de datos de entrada

En la figura 3-10 se muestran las distintas etapas del proceso de ejecución.

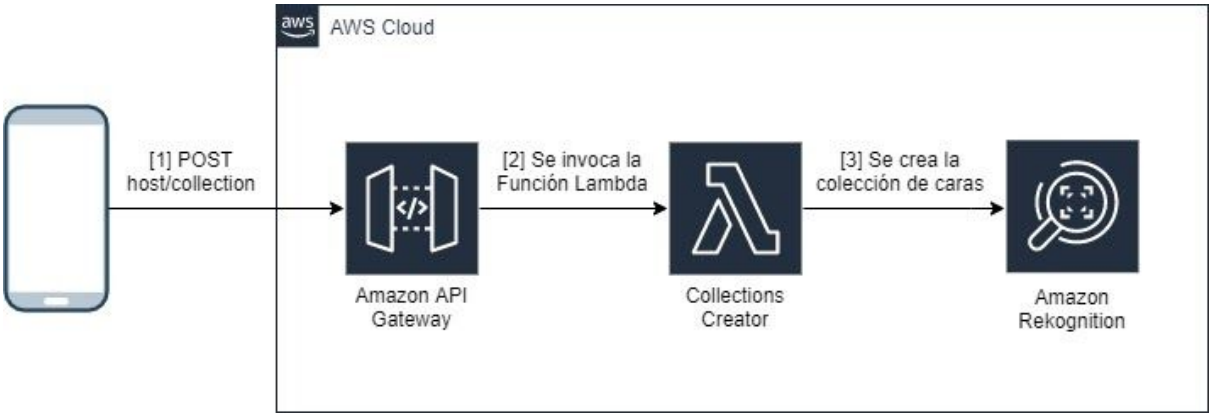

Figura 3-10. Lambda Collection Creator - Flujo de ejecución

El proceso comienza a través de la ejecución de una petición desde la aplicación móvil al endpoint *{host}/collection*. A partir de esta llamada a API Gateway se dispara la ejecución de la lambda Collections Creator. Esta lambda se comunica con Amazon Rekognition para crear la colección de caras. Una vez creada la colección, se almacena en la tabla Collections Table.

Faces uploader

Esta pieza del microservicio provee a la aplicación de una interfaz que le permite subir fotos de personas para ser registradas en el sistema.

Esta lambda se invoca a través de un evento producido por API Gateway cuando se realiza una petición de tipo post al endpoint *{host}/face*. Cuando es invocada, se extrae una imagen contenida en el body de la petición realizada y se sube al bucket S3 de la aplicación en la ruta */app/faces*. Esta subida de la imagen produce finalmente que se dispare la lambda Face Indexer.

Para ejecutar esta lambda es necesario proveerla de los siguientes parámetros de entrada.

Nombre del parámetro	Descripción
Token	Este parámetro es un JWT (**Json Web Token**). Esta ristra aparte de ser necesaria para la autenticación, contiene información acerca de la cuenta del usuario que está realizando la petición. Utilizando esta información somos capaces de acceder al conjunto de eventos asociados al usuario.
Client	Este parámetro nos permite obtener datos del cliente que ejecuta la petición. Esta información es guardada en los metadatos de la imagen subida. De esta forma podemos distinguir si la imagen se subió desde la app, o desde cualquier otro sitio.
Face	Este parámetro es un diccionario que contiene la información de la cara subida. Está compuesto por el nombre de la persona a la que sacamos la foto, la imagen y la orientación (frontal, frontal-derecha y frontal-izquierda)

Tabla 3-11. Lambda Faces Uploader - Tabla de datos de entrada

En la figura 3-11 se muestran las distintas etapas del proceso de ejecución.

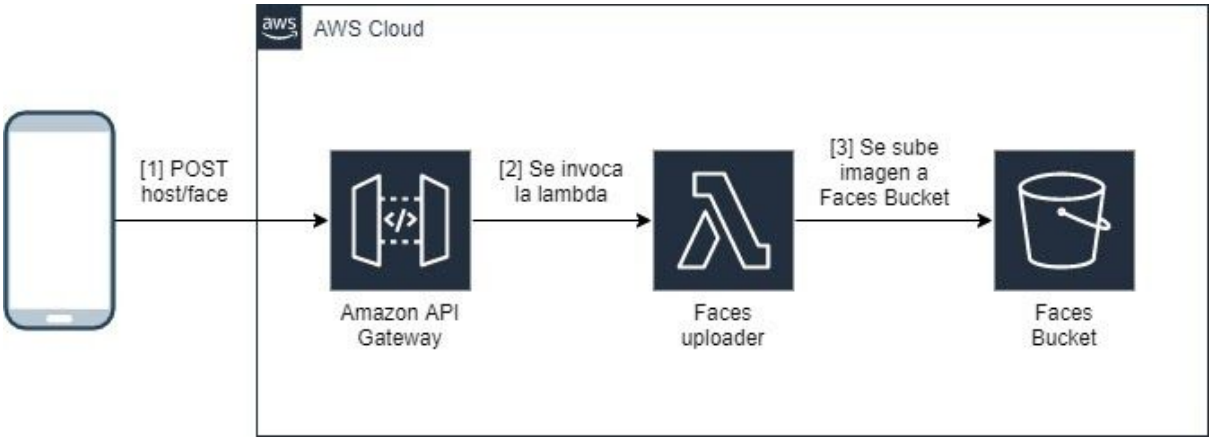

Figura 3-11. Lambda Faces Uploader - Flujo de ejecución

Faces indexer

Esta Lambda se encarga del proceso de registro de nuevas caras en el sistema. Se dispara a partir de un evento generado en S3, producido cada vez que se sube una imagen jpg a la ruta *app/faces/*. Cuando la lambda es invocada, extrae del evento la imagen y el nombre de la persona a registrar de los metadatos. Con esta imagen la lambda hace uso del servicio Amazon Rekognition para realizar un análisis y recibe un id asociado al resultado del proceso. Los datos extraídos del análisis se almacenan en la tabla Collection Table, creando un nuevo registro en la tabla, con el id obtenido del resultado del análisis de Amazon Rekognition y el nombre de la persona a registrar.

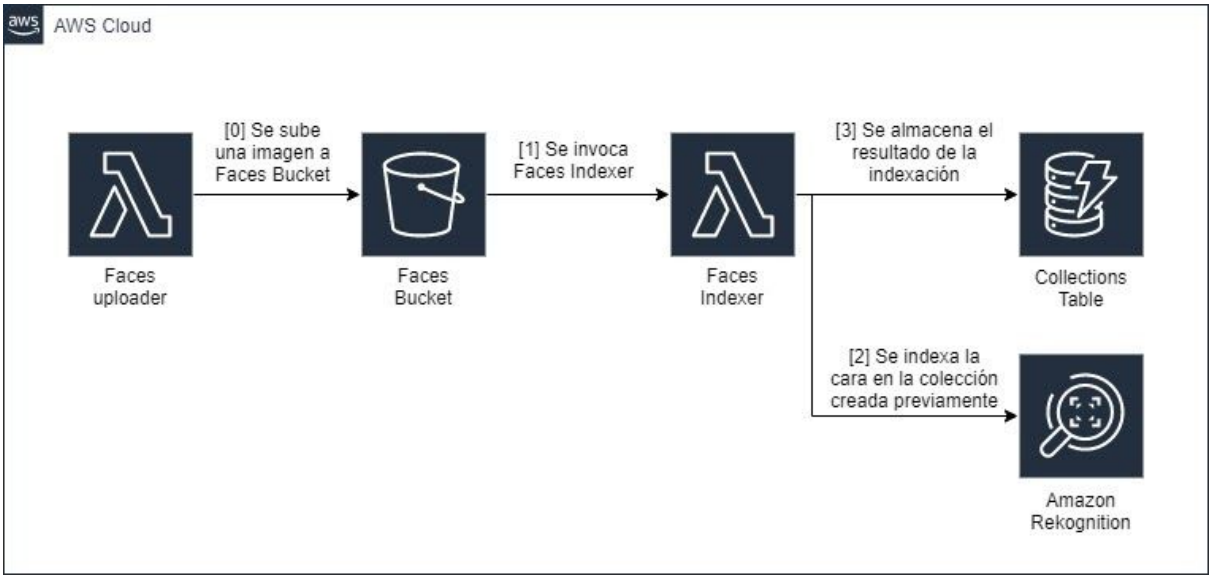

Figura 3-12. Lambda Faces Indexer - Flujo de ejecución

3.2.3 Analyzer Service

Este servicio se encarga de procesar las distintas capturas realizadas por la cámara y llevar a cabo un análisis facial de estas. Para llevar a cabo el reconocimiento facial de las imágenes, el servicio hace uso de la herramienta Amazon Rekognition. Esta herramienta forma parte del conjunto de recursos de Amazon Web Services utilizado.

A continuación se muestra el diseño utilizado en la implementación del servicio.

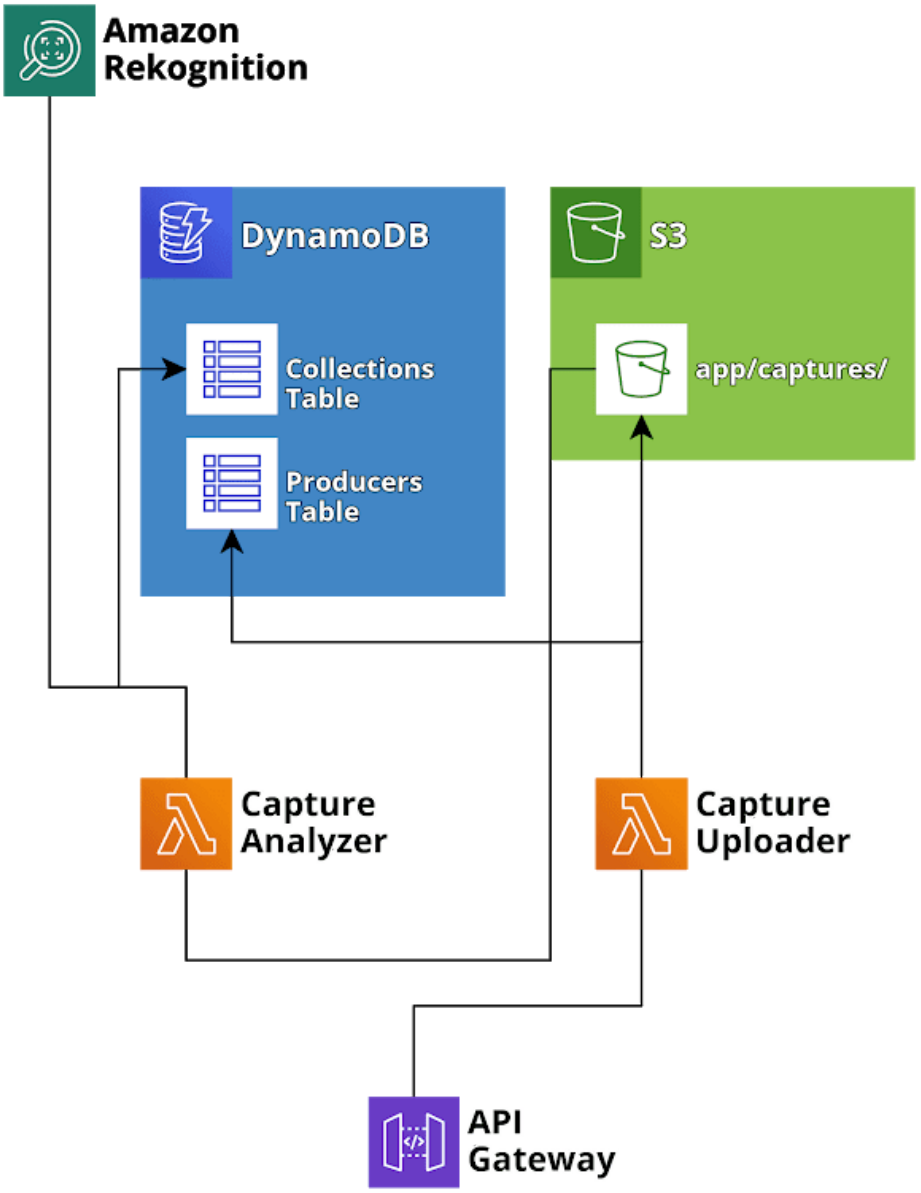

Figura 3-13. Analyzer Service - Arquitectura de servicio

Este servicio nos provee de dos funcionalidades, implementadas cada una con una función lambda propia. Estas funcionalidades son las representadas en la tabla 3-12.

Funcionalidad	Función lambda
Subir capturas realizadas por los productores	Capture Uploader
Analizar las capturas subidas por los productores	Capture Analyzer

Tabla 3-12. Analyzer Service - Tabla de funcionalidades

Captures Uploader

Esta función lambda provee a los distintos productores de la interfaz necesaria para subir imágenes a nuestro bucket en S3. Esta subida se hace mediante una petición http tipo post al endpoint *{host}/capture*. Cuando es invocada se extrae una imagen contenida en el body de la petición realizada y se sube al bucket S3 de la aplicación en la ruta */app/captures*. Esta subida de la imagen produce finalmente que se dispare la lambda Capture Analyzer.

La subida de capturas está condicionada al estado del productor. Para consultar el estado del productor, la lambda accede a la tabla de productores y consulta el valor del parámetro *state* del productor en cuestión. Si el valor de este parámetro es *'on'* se lleva a cabo la subida y sí es *'off'* no.

Cuando se ejecuta la función, debe cumplirse que existe en la tabla de productores un productor con el nombre indicado en la petición por el parámetro producer y con el secreto pasado por el parámetro secret. De no cumplirse estos requisitos la subida no se lleva a cabo.

Para llevar a cabo esta subida, es necesario proveer a la lambda de los siguientes parámetros de entrada.

Nombre del parámetro	Descripción
Token	Este parámetro es un JWT (**Json Web Token**). Esta ristra aparte de ser necesaria para la autenticación, contiene información acerca de la cuenta del usuario que está realizando la petición. Utilizando esta información somos capaces de acceder al conjunto de eventos asociados al usuario.

Nombre del parámetro	Descripción
Account	La cuenta del usuario que está subiendo la captura. Un ejemplo es usuario@gmail.com.
Producer	El nombre del productor que está subiendo la captura.
Capture	La imagen capturada en formato base64.
Date	Fecha en la que la captura fue realizada.
Secret	Secreto del productor. Esto nos permite autenticar al productor en el sistema del usuario.

Tabla 3-13. Captures Uploader - Tabla de parámetros

Como resultado de la ejecución de la lambda se obtiene el estado actual del productor, es decir, si está activado o no. En caso de no existir un productor con el nombre y secreto indicado se obtiene un error.

A continuación se muestra un diagrama representando el flujo del proceso de ejecución de esta función lambda.

En la figura 3-14 se muestran las distintas etapas del proceso de ejecución.

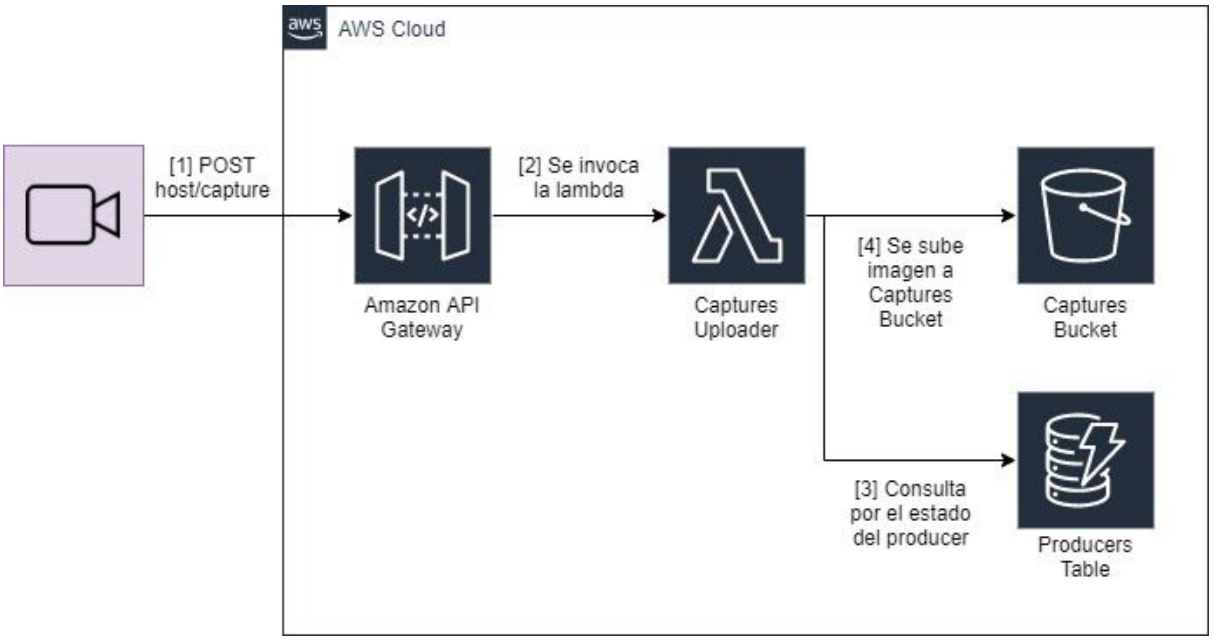

Figura 3-14. Lambda Captures Uploader - Flujo de ejecución

Captures Analyzer

Este componente es una función Lambda. Esta función lambda tiene como objetivo analizar las capturas que llegan al sistema a través de las distintas cámaras. Este proceso se ejecutará cuando se suba un archivo jpg a la ruta *app/captures* de nuestro bucket en S3. Al ejecutarse, la lambda, hace uso del servicio Amazon Rekognition para analizar la imagen llevando a cabo un reconocimiento de las caras que se encuentran en la misma. La lambda hace uso de la tabla Collections Table para poder relacionar las caras detectadas con las caras indexadas en el sistema. Esta relación se establece únicamente si la cara ha sido registrada.

Como resultado de la ejecución de esta lambda se genera un evento, el cual se guarda en la ruta */app/events*. El nombre utilizado para almacenar un evento se corresponde con el formato *{cuenta}/{productor}/{fecha}/{hora}/{segundo}.json.*

El evento creado contiene la información indicada en la tabla 3-14.

Campo	Descripción
Identificador de caras	Este campo es el identificador otorgado por amazon Rekognition para la cara reconocida en el proceso de análisis.
Nombre	Nombre correspondiente a la cara reconocida. Este campo no existe sí la cara no coincide con ninguna del diccionario de caras creado. El nombre se extrae de la tabla de colecciones.
Identificador de la colección	Este campo se corresponde con la colección sobre la que se ha hecho este análisis, es decir, del usuario de la aplicación.

Tabla 3-14. Captures Analyzer - Tabla de parámetros

En la figura 3-15 se muestran las distintas etapas del proceso de ejecución.

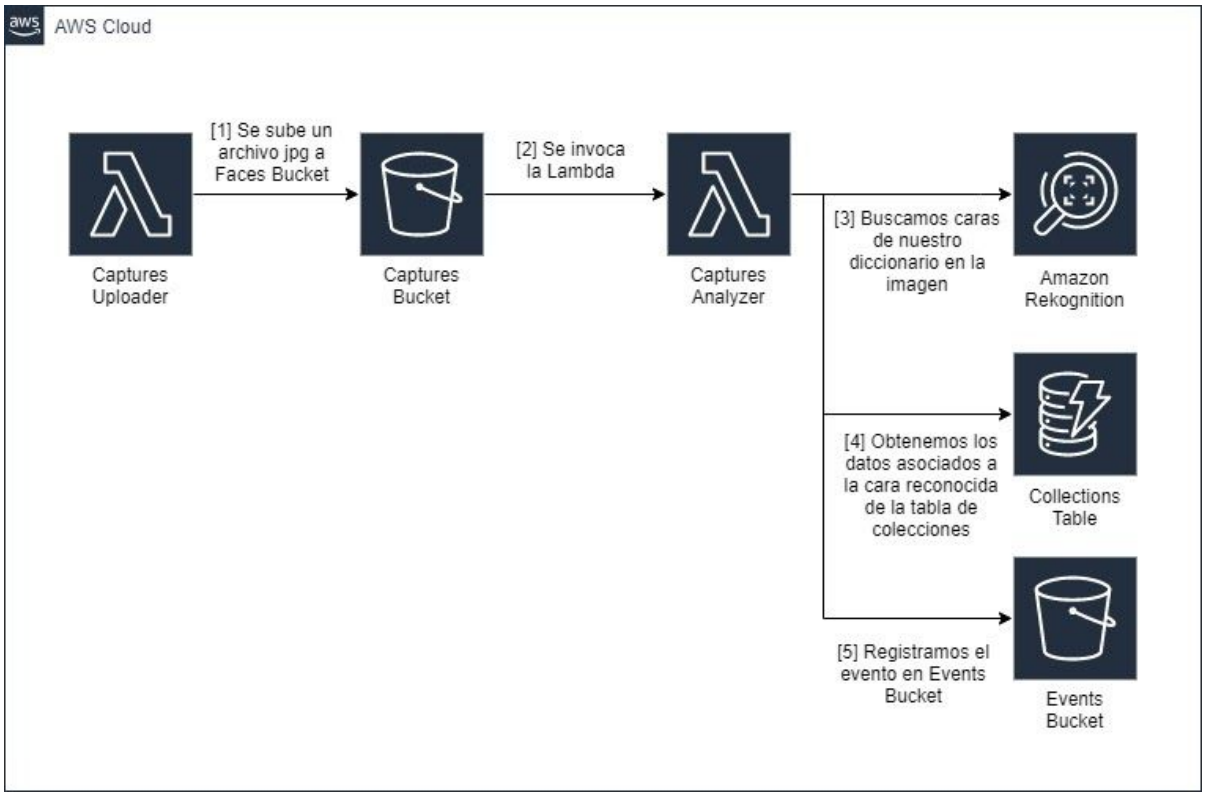

Figura 3-15. Flujo de ejecución - Lambda Captures Analyzer

3.2.4 Event Service

Para entender este servicio debemos conocer lo que es un evento dentro de nuestro sistema. **Denominamos evento al resultado del análisis y procesamiento de una captura tomada por una cámara registrada en el sistema**. El servicio event service se encarga de proveer a la aplicación móvil de nuestro sistema de la interfaz de comunicación necesaria para obtener los eventos.

La figura 3-16 muestra el diseño utilizado en la implementación del servicio.

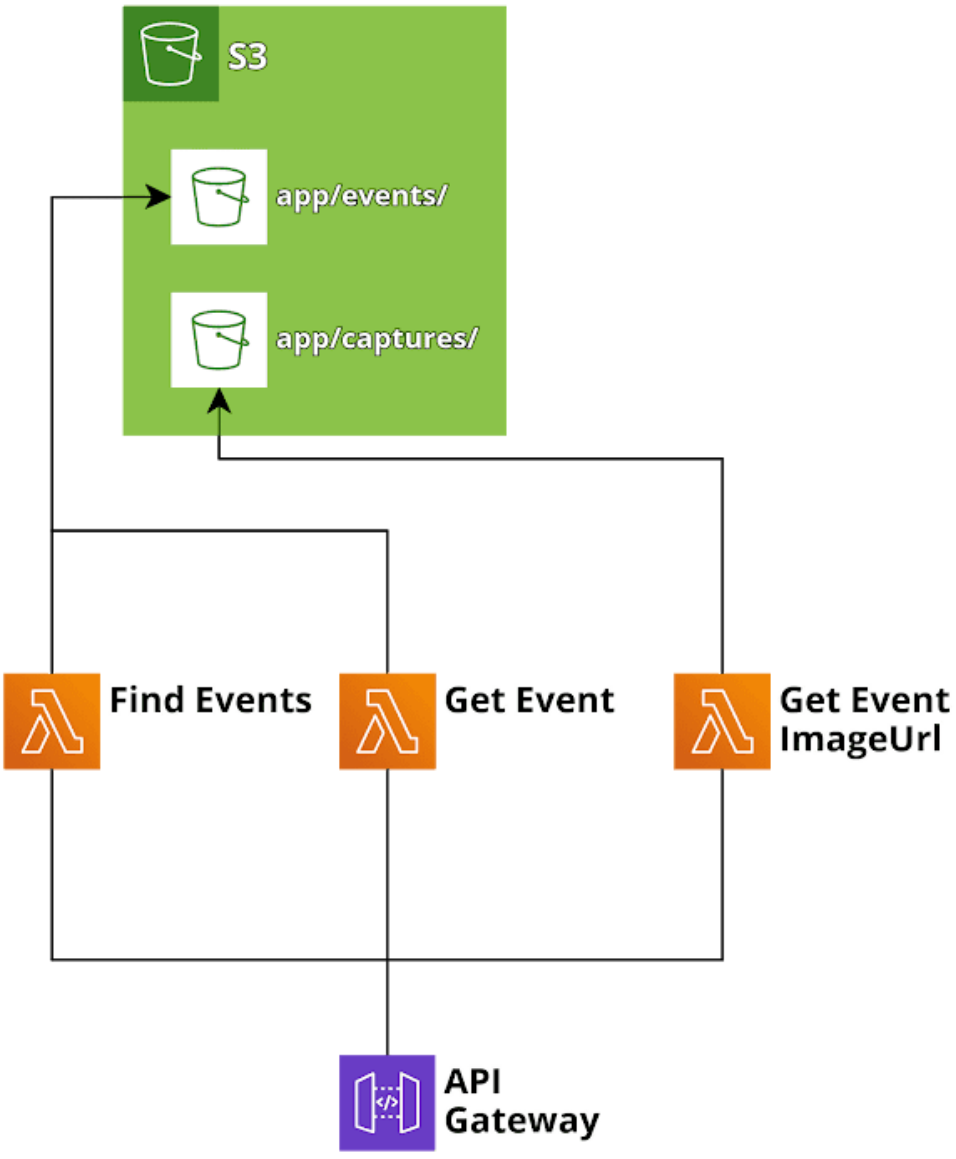

Figura 3-16. Event Service - Arquitectura de servicio

En concreto este servicio nos provee de tres funcionalidades, implementadas cada una con una función lambda propia. Estas funcionalidades son las representadas en la tabla 3-15.

Funcionalidad	Función lambda
Obtener listado de los identificadores de los eventos a mostrar en la aplicación móvil	FindEvents
Obtener el resultado del análisis de un evento	GetEvent
Obtener la captura de un evento	GetEventImageUrl

Tabla 3-15. Event Service - Tabla de funcionalidades

Find Events

Esta función lambda devuelve un array de identificadores de los eventos asociados a un sistema, los cuales pueden pertenecer a distintas cámaras. La función en cuestión se utiliza para mostrar los últimos eventos registrados por el sistema en la aplicación. Debido a limitaciones en el tamaño de las respuestas de las lambdas y con el objetivo de optimizar el consumo de datos de la aplicación, el número de eventos devueltos está limitado.

Los parámetros utilizados para llamar a esta función son los definidos en la tabla 3-16.

Nombre del parámetro	Descripción
Continuation Token	Cuando ejecutamos una petición a la función, en la respuesta se nos pasa un *continuation token*, pasando este parámetro en la siguiente petición que hagamos a este endpoint podremos indicar a partir de que evento debemos realizar la búsqueda. Para entender este funcionamiento hay que tener en cuenta que los eventos se encuentran ordenados por fecha. Este parámetro es opcional y si no se devuelve se comienza por el primero de los eventos.
Max Keys	Este parámetro opcional nos permite indicar el número máximo de claves de eventos a devolver por la función.

Nombre del parámetro	Descripción
Day y minute	Estos parámetros opcionales nos permiten obtener eventos asociados a una fecha.
Token	Este parámetro es un JWT (**Json Web Token**). Esta ristra aparte de ser necesaria para la autenticación, contiene información acerca de la cuenta del usuario que está realizando la petición. Utilizando esta información somos capaces de acceder al conjunto de eventos asociados al usuario.
Producer	Este parámetro es opcional y nos permite obtener los eventos asociados a un productor en concreto.

Tabla 3-16. Find Events - Tabla de parámetros

En la figura 3-17 se muestran las distintas etapas del proceso de ejecución.

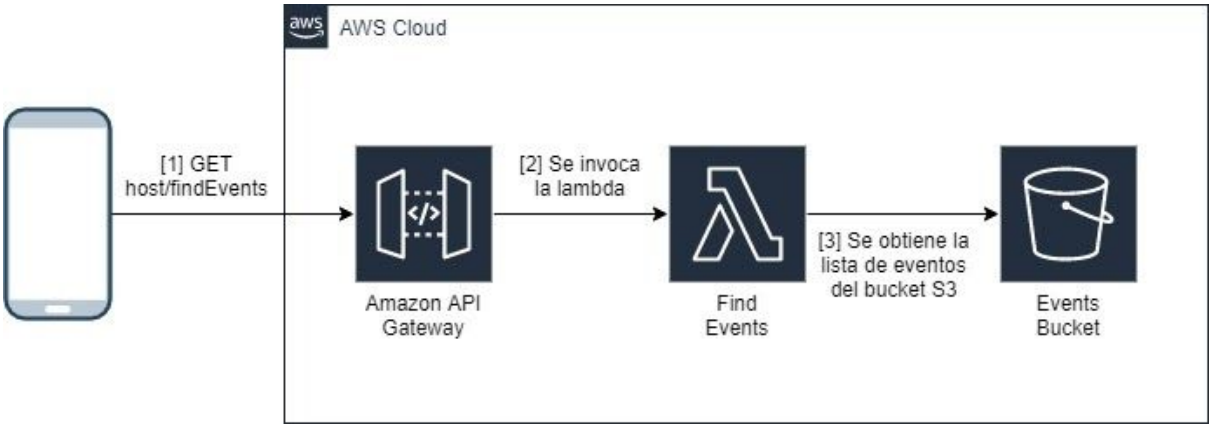

Figura 3-17. Flujo de ejecución - Lambda Find Events

Get Event

Esta función nos permite obtener un evento determinado a partir de su clave. Esta función se utiliza en la aplicación para mostrar los detalles de un evento desde la vista de detalles del evento. El único parámetro de entrada necesario para ejecutar esta lambda es la clave del evento a obtener, para pasar esta clave utilizamos el parámetro *object_key*.

La información devuelta por esta función lambda se corresponde con los parámetros mostrados en la tabla 3-17.

Nombre del parámetro	Descripción
ImageUrl	Url para descargar la imagen desde la aplicación. Esta url es válida durante tres minutos.
Producer	El productor que capturó la imagen.
Date	Fecha y hora en la que se tomó la imagen.
Recognition	Contiene la información acerca de datos reconocidos en la imagen. En caso de reconocer a alguien correctamente el nombre.
ObjectKey	La clave del objeto que representa el evento en S3. La misma clave pasada al ejecutar la petición.

Tabla 3-17. Get Event- Tabla de parámetros

En la figura 3-18 se muestran las distintas etapas del proceso de ejecución.

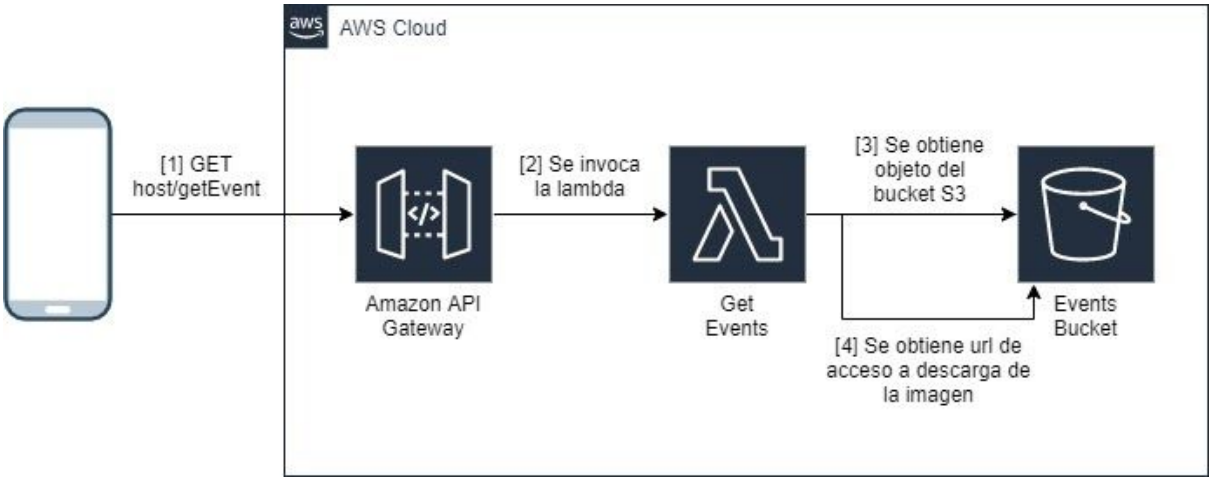

Figura 3-18. Flujo de ejecución - Lambda Get Event

3.2.5 Controller Service

Este servicio pretende ofrecer a los usuarios la capacidad de activar y desactivar el procesamiento de las capturas realizadas por las cámaras. Esto nos permite reducir los costes asociado al procesamiento de imágenes sin dejar de tomar capturas. Con este fin, este servicio provee a la aplicación móvil de una API con los endpoints necesarios para cumplir con el objetivo del servicio.

La figura 3-19 expuesta a continuación muestra el diseño utilizado en la implementación del servicio.

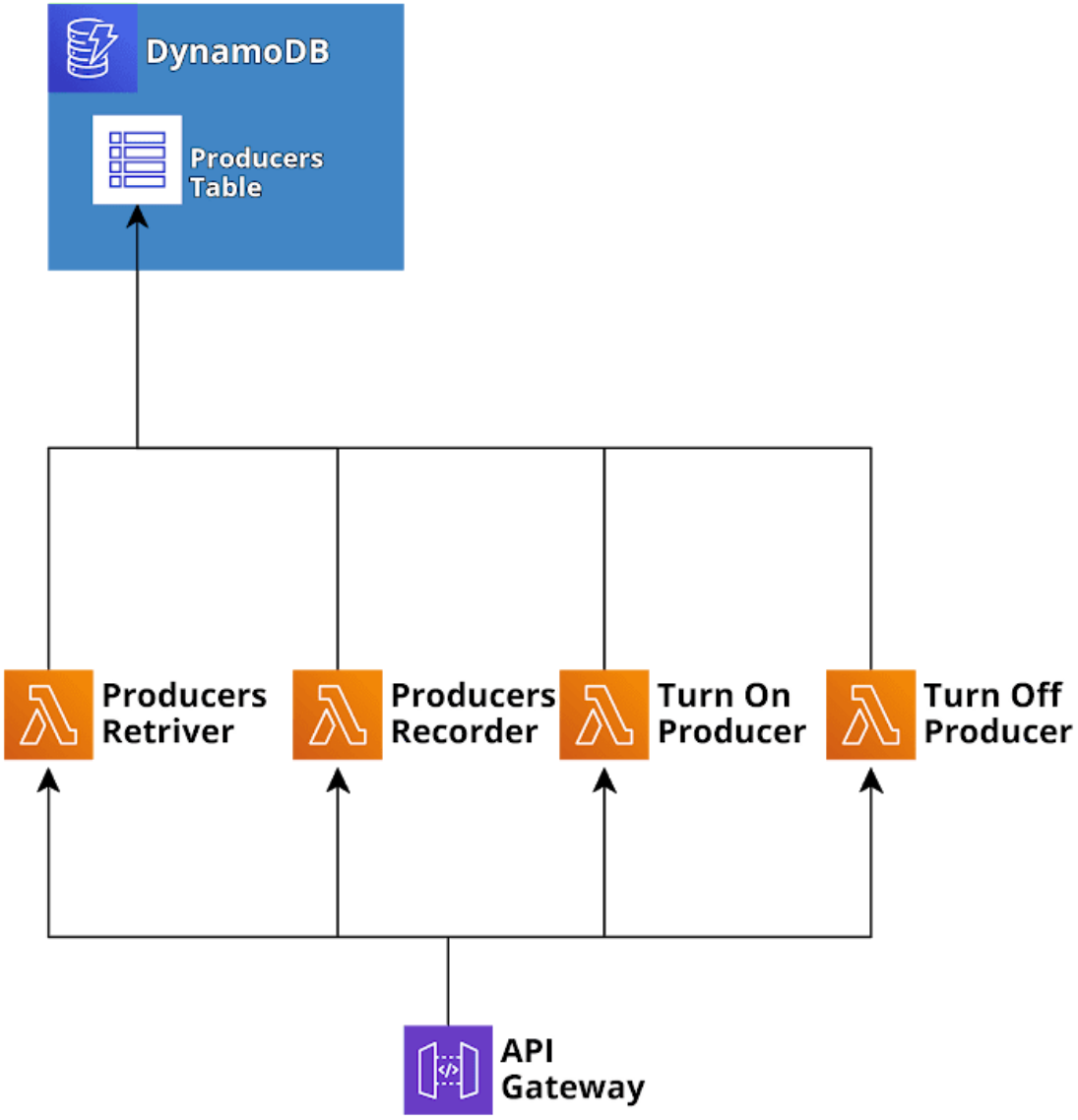

Figura 3-19. Controller Service - Arquitectura de servicio

En concreto, este servicio nos provee de tres funcionalidades, implementadas cada una con una función lambda propia. Estas funcionalidades son las representadas en la siguiente tabla.

Funcionalidad	Función lambda
Obtener listado de los productores registrados en el sistema de un usuario concreto	Producers Retriever
Registrar un nuevo productor en un sistema de un usuario	Producers Recorder
Encender un productor	Turn On Producer
Apagar un productor	Turn Off Producer

Controller Service - Tabla de funcionalidades

Producers Retriever

Esta función lambda devuelve el conjunto de productores registrados en una cuenta, para ello hace uso de la tabla de productores. Esta tabla contiene información acerca de los productores y su estado. De esta forma no solo obtenemos un listado de los productores de la cuenta sino que somos capaces de saber si están encendidos o apagados.

Los parámetros utilizados para llamar a esta función son los definidos en la tabla *Tabla 3-18*.

Nombre del parámetro	Descripción
Token	Este parámetro es un JWT (**Json Web Token**). Esta ristra, aparte de ser necesaria para la autenticación, contiene información acerca de la cuenta del usuario que está realizando la petición. Utilizando esta información somos capaces de acceder al conjunto de productores del usuario.

Tabla 3-18. Producers Retriever - Tabla de parámetros

En la figura 3-20 se muestran las distintas etapas del proceso de ejecución.

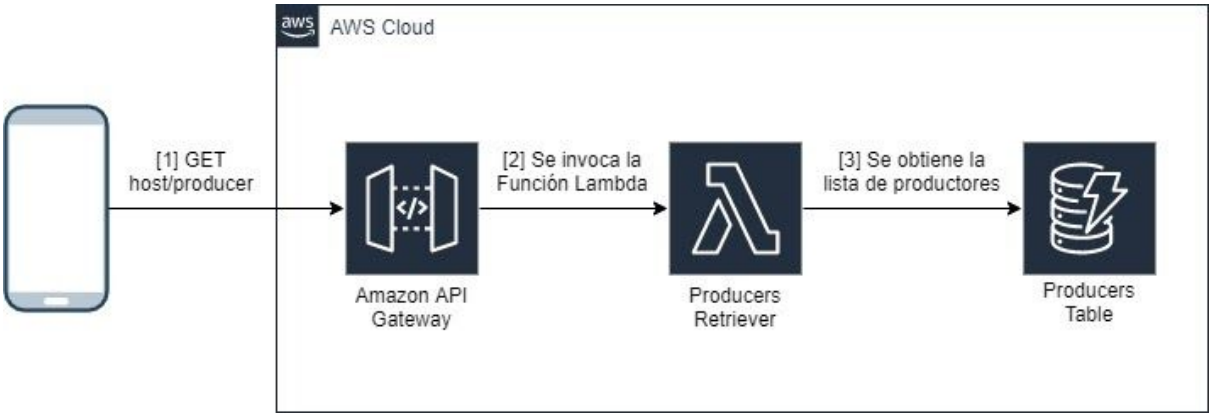

Figura 3-20. Flujo de ejecución - Lambda Producers Retriever

Producers Recorder

Esta función se encarga de registrar nuevos productores en una cuenta. Para ello hace uso de la tabla de productores. Cuando ejecutamos esta lambda a través de una petición POST al endpoint *{host}/producer* se crea un nuevo registro en la tabla de productores. Al hacer esto, se establece el nombre y el secreto que deberá tener el productor que suba las imágenes al sistema a través del *Analyzer Service*.

Esta función lambda nos permite también modificar los datos de un productor, esto se produce cuando el productor ya está registrado.

Nombre del parámetro	Descripción
Token	Este parámetro es un JWT (*Json Web Token*). Esta ristra, aparte de ser necesaria para la autenticación, contiene información acerca de la cuenta del usuario que está realizando la petición. Utilizando esta información somos capaces de acceder al conjunto de productores del usuario.
Name	Nombre del productor. Este es un nombre único, y el productor deberá tener esta información y utilizarla cuando realice la subida de una captura.
Secret	Secreto asociado al productor. Es usado por el sistema como elemento autenticación para subir imágenes desde un productor.

Tabla 3-19. Producers Recorder - Tabla de parámetros

En la figura 3-21 se muestran las distintas etapas del proceso de ejecución.

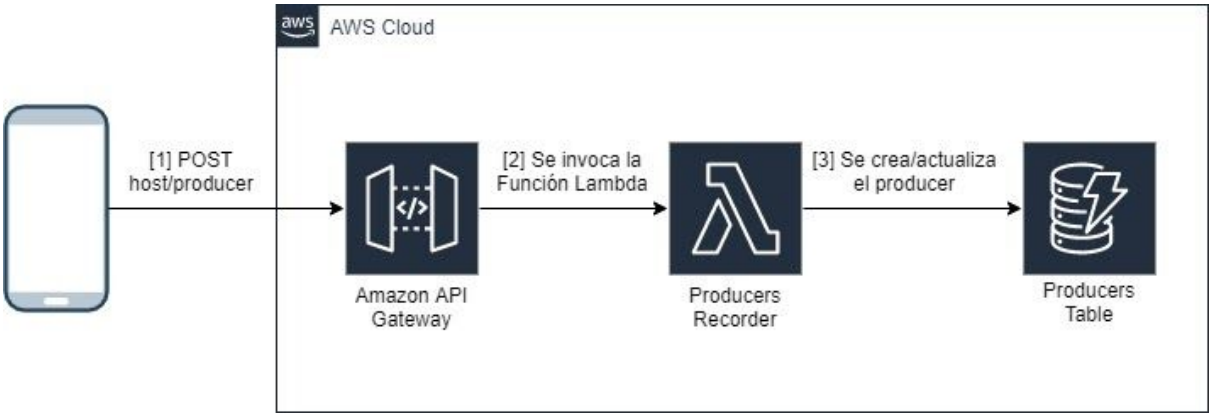

Figura 3-21. Flujo de ejecución - Lambda Producers Recorder

Turn On Producer

Esta función se encarga de encender los distintos productores. Para llevar a cabo esta tarea modifica el estado de los productores en la tabla de productores. Cuando ejecutamos esta función sobre un productor cuyo campo state tenga valor *'off'*, este pasa a tener valor *'on'*. En el caso de que la ejecutemos sobre un productor cuyo estado es *'on',* este se mantiene en este estado.

El estado de un productor define si, a la hora de subir las capturas a través de la Lambda *Image Uploader* del servicio *Analyzer Service*, esta subida se lleva a cabo o si por el contrario no se permite. Esta funcionalidad nos permite limitar el procesamiento en Amazon Rekognition, dado que al no subir las imagenes, no se dispara la Lambda *Faces Analyzer*,

Los parámetros utilizados para llamar a esta función son los definidos en la tabla 3-19.

Nombre del parámetro	Descripción
Token	Este parámetro es un JWT (**Json Web Token**). Esta ristra, aparte de ser necesaria para la autenticación, contiene información acerca de la cuenta del usuario que está realizando la petición. Utilizando esta información somos capaces de acceder al conjunto de productores del usuario.
Producer	El nombre del productor a encender.

Tabla 3-19. Turn On Producer - Tabla de parámetros

En la figura 3-22 se muestran las distintas etapas del proceso de ejecución.

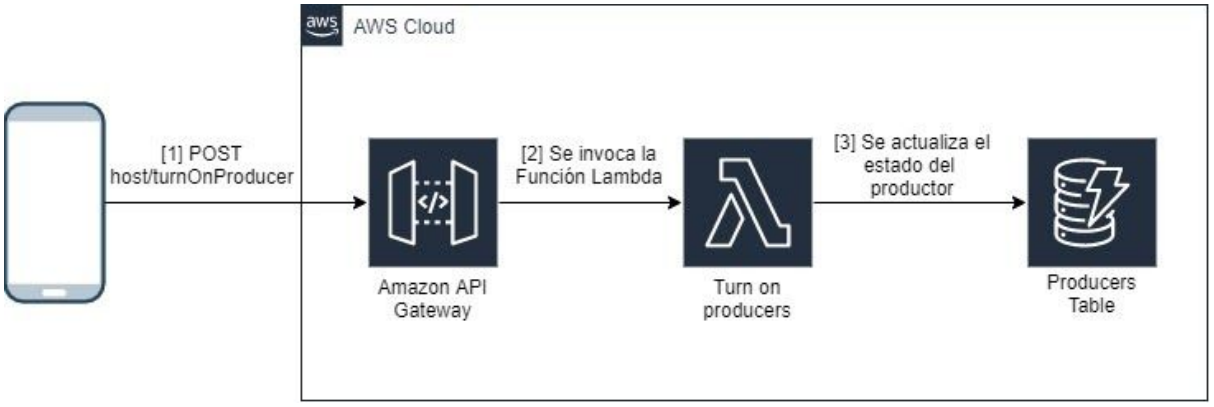

Figura 3-22. Flujo de ejecución - Lambda Producers Recorder

Turn Off Producer

Esta función se encarga de apagar los distintos productores. Para llevar a cabo esta tarea modifica el estado de los productores en la tabla de productores. Cuando ejecutamos esta función sobre un productor, cuyo campo state tenga valor *'on'*, este pasa a tener valor *'off'*. En el caso de que la ejecutemos sobre un productor cuyo estado es *'off',* este se mantiene en este estado.

El estado de un productor define si a la hora de subir las capturas a través de la Lambda *Image Uploader* del servicio *Analyzer Service*, esta subida se lleva a cabo o si por el contrario no se permite. Esta funcionalidad nos permite limitar el procesamiento en Amazon Rekognition, dado que al no subir las imagenes no se dispara la Lambda *Faces Analyzer*,

Los parámetros utilizados para llamar a esta función son los definidos en la tabla 3-20.

Nombre del parámetro	Descripción
Token	Este parámetro es un JWT (*Json Web Token*). Esta ristra aparte de ser necesaria para la autenticación, contiene información acerca de la cuenta del usuario que está realizando la petición. Utilizando esta información somos capaces de acceder al conjunto de productores del usuario.
Producer	El nombre del productor a encender.

Tabla 3-20. Turn Off Producer - Tabla de parámetros

En la figura 3-23 se muestran las distintas etapas del proceso de ejecución.

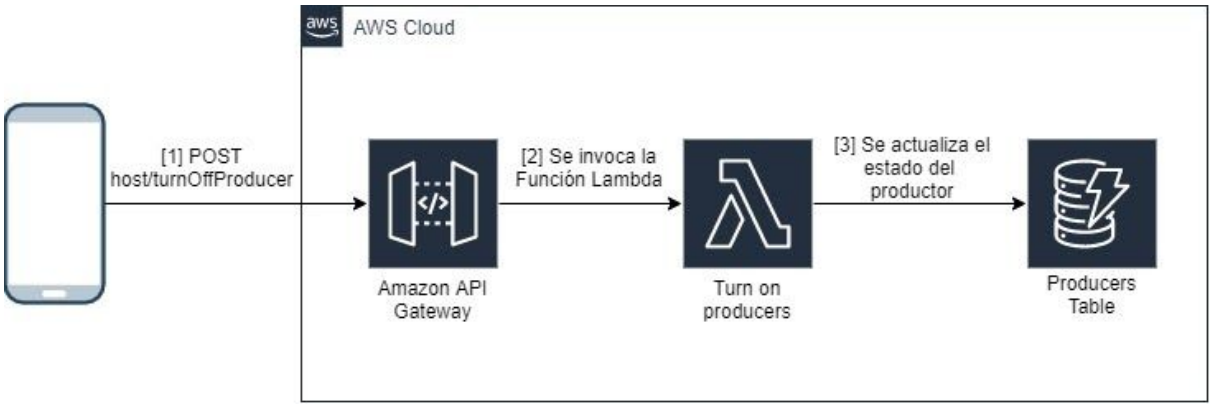

Figura 3-23. Flujo de ejecución - Lambda Producers Recorder

3.3 Diseño de la aplicación móvil

En este apartado se trata el diseño de la aplicación móvil implementada. Para ello se abordarán dos secciones, estas dos secciones son:

- Diseño de la interfaz de usuario
- Diseño del software

3.3.1 Diseño de interfaz de usuario

Paleta de colores

A continuación se muestra una tabla representando la paleta de colores y el uso de cada color. La tabla 3-21 representa los colores utilizados en la implementación de la aplicación y no los del prototipo planteado en los mockups.

Color	Uso
#FFFFFFFF	Utilizado para el fondo de pantalla de la vista de accesos y la vista de indexado de caras. También se utiliza para los iconos y botones.
#CFD8DC	Utilizado en en el contenedor de imágenes de la vista de indexación de caras, cuando no hay imágenes capturadas.
#FFB0BEC5	Utilizado en botones y como fondo en las tarjetas de accesos registrados una vez volteadas
#FF455A64	Utilizado para destacar texto sobre fondos de color #FFB0BEC5.
#FF37474F	Utilizado para destacar texto sobre fondos de color #FFB0BEC5.
#FF263238	Utilizado para la barra de navegación, la barra superior de la aplicación, para las tarjetas de la vista de accesos registrados, botones y para el fondo de la vista del controlador.
#FFFF8A80	Color utilizado para el fondo de las tarjetas de acceso de caras no conocidas en la vista de accesos

#FFFF5252	Color utilizado para notificar alerta por acceso de persona no indexada en la vista de accesos y para indicar que un productor está apagado

Tabla 3-21. Tabla de paletas de colores

Diseño de las vistas

El diseño de las vistas se corresponde en gran medida con el diseño de las vistas planteadas en la sección de mockups. A continuación se muestra el resultado obtenido para cada vista, su correspondiente finalidad y funcionamiento.

Vista de inicio de aplicación

Esta vista se corresponde con la imagen mostrada previamente a la apertura de la aplicación. A esta vista se le denomina *SplashScreen* y en este caso se compone de una imagen, un texto y un indicador.

Figura 3-24. Diseño de aplicación - Vista de inicio de aplicación

Vista de login

Esta vista se corresponde con el proceso de identificación de usuarios en la aplicación. En esta vista el usuario introduce sus credenciales y se identifica en la aplicación.

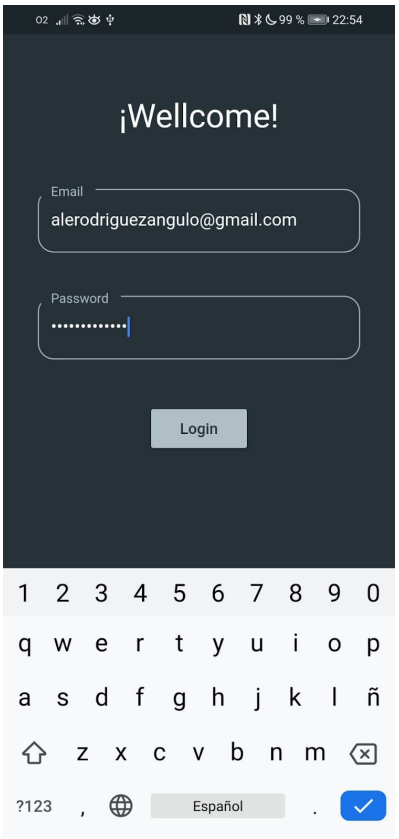

Figura 3-25. Diseño de aplicación - Vista de login

Vista de accesos registrados

Esta vista nos muestra el listado de accesos registrados por el sistema, además de información sobre estos.

La vista en cuestión tiene dos estados posibles. El primer estado se asocia a cuando no hay ningún acceso registrado. El segundo estado se corresponde con una situación en la que sí hayan accesos registrados.

Por defecto, en cada uno de los elementos de la lista de accesos podemos ver el productor que registró el acceso, la fecha del acceso y es un acceso permitido o no. Para obtener más información sobre un acceso registrado por el sistema, el usuario debe pulsar sobre la tarjeta asociada a dicho acceso, esta se volteara y le mostrará la persona que se ha identificado o en su defecto un mensaje que indica que es un acceso no permitido. Cuando

se voltea una tarjeta esta muestra un botón de imágenes, al pulsar este botón accedemos a la *Vista de visualización de imágenes acceso registrado*.

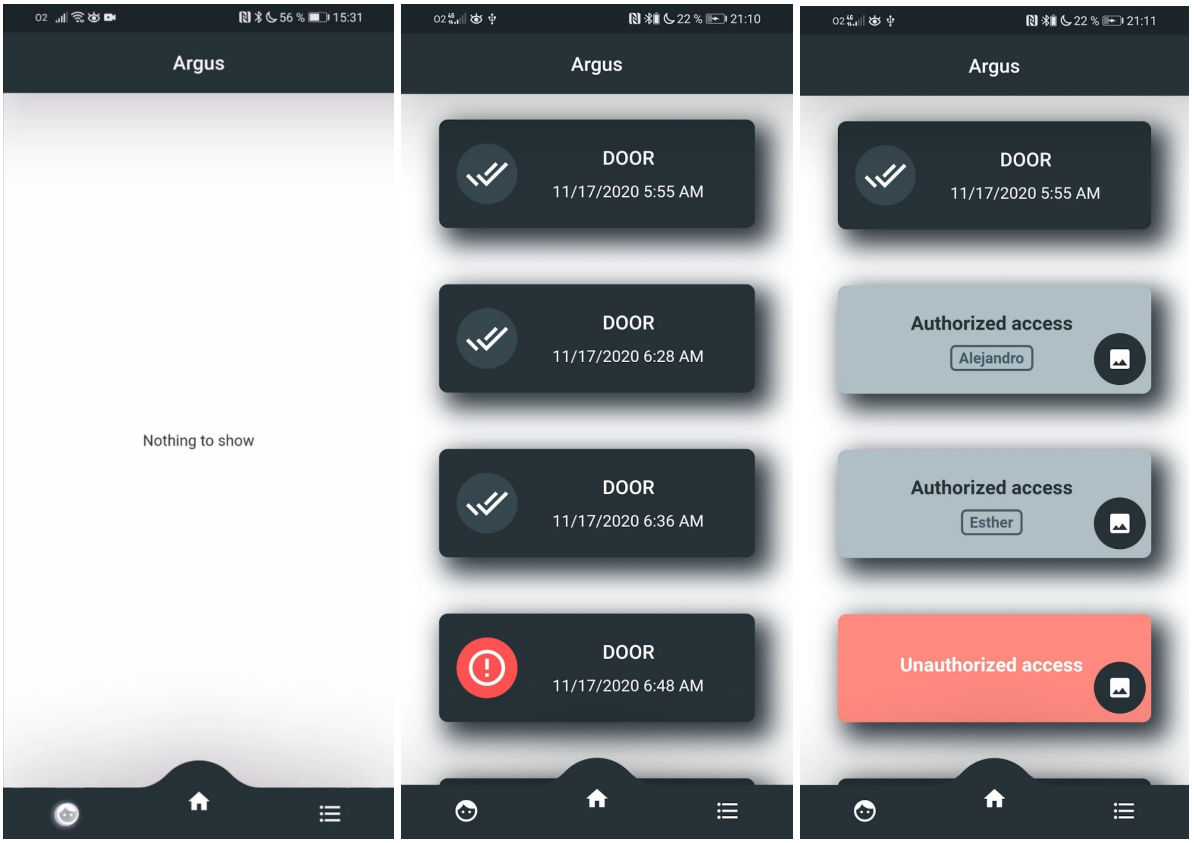

Figura 3-26. Diseño de aplicación - Vista de visualización de imágenes acceso registrado

Vista de visualización de imágenes acceso registrado

En esta vista se muestran las imágenes capturadas como resultado del registro de un acceso. Por defecto estas imágenes se alternan automáticamente para hacer más sencilla la visualización.

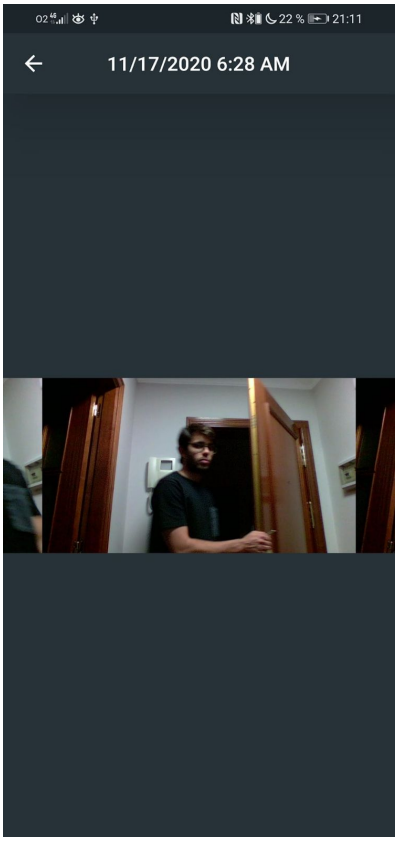

Figura 3-27. Diseño de aplicación - Vista de visualización de imágenes acceso registrado

Vista de indexado de caras

Esta vista nos permite realizar el indexado de caras. Para ello nos provee de dos funcionalidades, la primera sacar fotos y la segunda un formulario para introducir el nombre de la persona a registrar. Cuando se sacan tres fotos, se habilita al usuario a ejecutar el registro de caras pulsando el botón correspondiente.

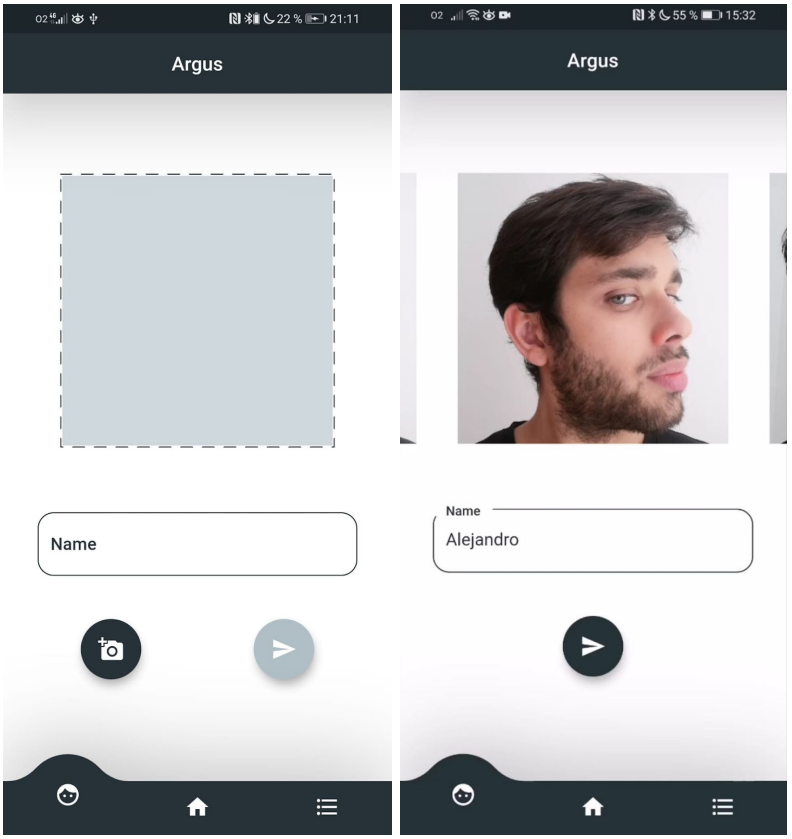

Figura 3-28. Diseño de aplicación - Vista Vista de indexado de caras

Vista del controlador

Esta vista nos permite apagar, encender y registrar productores. Para ello nos muestra una lista de productores, cuyo estado podemos modificar haciendo uso del botón de apagado y de un botón para el registro de nuevos productores. Cuando queremos registrar un productor se nos pregunta por el nombre del mismo y el secreto.

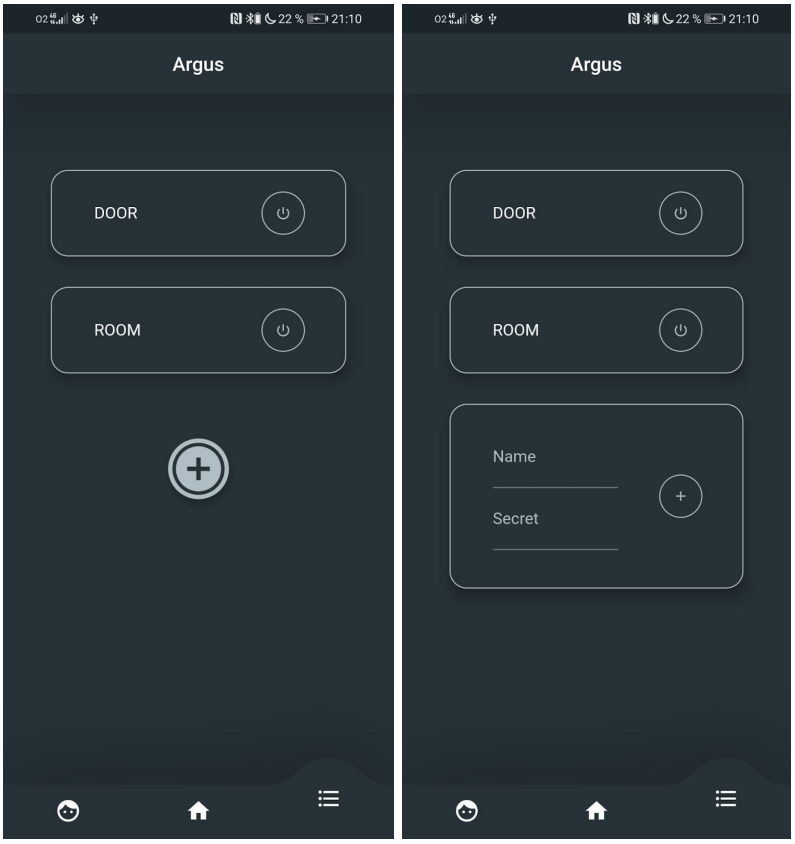

Figura 3-29. Diseño de aplicación - Vista del controlador

3.3.2 Organización del software y piezas software

Organización del código

El código se organiza en base a las recomendaciones de los desarrolladores de Flutter Framework. A lo largo del código se trata de seguir el patrón Model View Controller o MVC. A continuación tratamos de explicar cada uno de los archivos y carpetas principales que nos encontramos.

Archivo main.dart

Este archivo es el punto de entrada a la aplicación. En el inicializamos los distintos servicios que hemos creado dentro de la carpeta *src/controller/services* y posteriormente lanzamos la aplicación.

Carpeta src /

Esta carpeta contiene todos los archivos relativos a la aplicación. Se encuentra dividido en subcarpetas de acuerdo con el patrón Model View Controller o MVC.

Nombre del Servicio	Descripción
controller	Contiene servicios y clientes de la aplicación
model	Contiene las clases que modelan los datos de la aplicación.
ui	Contiene todo el software relativo a la interfaz de usuario de la aplicación.

Tabla 3-22. Diseño de aplicación móvil - Tabla de contenidos de carpeta src

Piezas software

A continuación tratamos de describir las funcionalidades de las piezas de software desarrolladas. Las abordaremos en función del tipo de pieza que son.

Servicios

Como parte de la aplicación móvil se plantea desarrollar un conjunto de servicios. Estos servicios son piezas software que nos proveen de un conjunto de funcionalidades concretas.

A continuación se muestra la tabla 3-23 que describe los servicios a desarrollar.

Nombre del Servicio	Descripción
Servicio de autenticación	Este servicio nos provee de todas las funcionalidades necesarias para llevar a cabo la autenticación del usuario.
Servicio del controlador	Este servicio nos provee de todas las funcionalidades necesarias para interactuar con el Controller Service desarrollado como parte de la sección de infraestructura.

Nombre del Servicio	Descripción
Servicio de eventos	Este servicio contiene todas las funcionalidades necesarias para interactuar con el Event Service desarrollado como parte de la sección de infraestructura.
Servicio de indexado	Este servicio contiene todas las funcionalidades necesarias para interactuar con el Indexer Service desarrollado como parte de la sección de infraestructura.
Servicio de almacenamiento	Este servicio nos provee de todas las funcionalidades necesarias para llevar a cabo el almacenado de datos en el móvil en el que se ejecuta la aplicación.

Tabla 3-23. Diseño de aplicación móvil - Tabla de servicios

Clientes

Para dar soporte a algunos de los servicios planteados, es necesario el desarrollo de una serie de clientes. Un cliente en este contexto, es una pieza software que nos provee de la interfaz de comunicación necesaria para comunicarnos con un servicio web.

La utilización de este tipo de piezas nos permite encapsular la lógica asociada al acceso a recursos web, haciéndola más mantenible y siguiendo los principios SOLID.

Nombre del Cliente	Descripción
Cliente de autenticación	Este cliente actúa como una interfaz de comunicación con los recursos asociados a la autenticación de usuarios.
Cliente del controlador	Este cliente actúa como una interfaz de comunicación con los recursos asociados al Controller Service.
Cliente de eventos	Este cliente actúa como una interfaz de comunicación con los recursos asociados al Event Service.

Nombre del Cliente	Descripción
Cliente de indexado	Este cliente actúa como una interfaz de comunicación con los recursos asociados al Client Service.

Tabla 3-24. Diseño de aplicación móvil - Tabla de clientes

Widgets

Un widget es el elemento básico de flutter. En Flutter todo lo que creamos son widgets. Un botón, una tarjeta, una barra de opciones o incluso una aplicación en sí misma, son Widgets.

Dentro de la carpeta *src/ui/* encontraremos una serie de widgets, estos widgets se distribuyen entre dos subcarpetas, las cuales son *screens/* y *widgets/*. La primera de estas carpetas contiene una serie de widgets que se corresponden con las distintas screens desarrolladas. La segunda carpeta se corresponde con un conjunto de widgets utilizados en las distintas vistas.

Cabe destacar la existencia de dos archivos, *schedule.dart* y *themeProvider.dart* de especial importancia. En el primero se corresponde con el *Patrón Provider* utilizado para el control de estados en la aplicación, en el segundo define el tema de la aplicación.

4. Tecnologías y herramientas

A lo largo de este apartado se abordan las tecnologías utilizadas así como patrones o prácticas seguidas, describiendolas y justificando su uso. Estas tecnologías se encuentran organizadas en función de las piezas implementadas en las que fueron utilizadas.

4.1 Productores

Las tecnologías utilizadas para el desarrollo de los productores se dividen en dos categorías, las tecnologías asociadas al desarrollo del hardware y las tecnologías asociadas al desarrollo del software. A continuación se describe cada una de las tecnologías utilizadas.

Raspberry PI 4

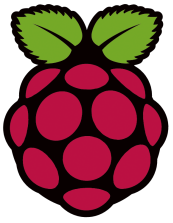

Figura 4-1. Logo de RaspBerry Pi [14]

La Raspberry Pi es un ordenador de dimensiones muy reducidas. Este ordenador cuenta con todos los componentes hardware de un ordenador de sobremesa común, o en su defecto algún sustitutivo. Con este elemento creamos el hardware de la cámara, para ello necesitamos de algunos componentes adicionales, como una cámara, una tarjeta de memoria y el cableado.

Justificación de elección: Hay dos razones principales por la que se utiliza este dispositivo y no alguno de sus competidores. La primera de estas razones es el fácil acceso a este dispositivo, podemos encontrar una Raspberry Pi 4 en cualquier tienda online e incluso en algunas tiendas locales. La segunda de estas razones, y no por ello menos importante, es la amplia comunidad de la que disponemos.

Sitio web: https://www.raspberrypi.org/

Docker

Figura 4-2. Logo de Docker [15]

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software. Proporcionan una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker utiliza características de aislamiento de recursos del kernel Linux, tales como cgroups y espacios de nombres (namespaces) para permitir que "contenedores" independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales. [15]

Justificación de elección: Los motivos por los que se eligió docker fueron varios. El primero fue su compatibilidad con Windows y Linux, esto hacía posible el desarrollo desde cualquier dispositivo. El segundo la gran comunidad y documentación que hay alrededor del proyecto. El tercero la enorme cantidad de imágenes creadas que hay disponibles de forma gratuita, algunas de las cuales utilizamos en este prototipo.

Sitio web: https://docs.docker.com

Docker Compose

Figura 4-3. Logo de Docker Compose [16]

Esta es una herramienta de definición y ejecución de aplicaciones Docker multicontenedor. Con esta herramienta podemos configurar nuestras aplicaciones y servicios Docker a través

de un archivo YAML. Nos ofrece control sobre las conexiones entre los contenedores y un conjunto de comandos útiles para el desarrollo y puesta en marcha de los servicios..

Justificación de elección: Se eligió esta herramienta principalmente por su compatibilidad con Docker. Además aunque es cierto que hay otras herramientas en el mercado similares, esta es la más extendida y por ende la que más comunidad tiene.

Sitio web: https://docs.docker.com/compose/

Java

Figura 4-4. Logo de Java [17]

Java es un lenguaje de programación orientado a objetos. En este caso se utiliza para crear una pieza de software que se encarga de la detección de movimiento a través de imágenes. Para ello se utiliza la tecnología openCV.

Justificación de elección: Se eligió utilizar este lenguaje porque disponía de una librería bastante completa (openCV). Aunque es cierto que existen alternativas como Python con esta misma herramienta disponible, se escogió Java debido a preferencias personales del autor.

Sitio web: https://www.java.com/es/

OpenCV

Figura 4-5. Logo de Java [18]

Librería basada en la visión por computador. Esta librería nos provee de diversas herramientas para tratar imágenes y vídeos. Utilizamos esta tecnología para la detección de movimientos a través de las cámaras.

Justificación de elección: Se eligió utilizar esta herramienta debido a que es opensource, la más utilizada en el mercado y a la experiencia previa obtenida en la universidad.

Sitio web: https://www.opencv.org

RabbitMQ

Figura 4-6. Logo de RabbitMQ [19]

RabbitMQ es un broker de mensajería. Este software nos permite crear colas y utilizarlas como medio de comunicación entre los contenedores creados.

Justificación de elección: Se eligió utilizar esta tecnología y no otro broker de mensajería presente en el mercado porque open source, además del standart en aplicaciones basadas en microservicios.

Sitio web: https://www.rabbitmq.com/

Python

Figura 4-7. Logo de Python [20]

Python es un lenguaje de programación basado en scripting. En este caso se utiliza para crear una pieza de software que se encarga de subir las imágenes al entorno cloud cuando son capturadas por el proceso descrito anteriormente.

Justificación de elección: Se eligió utilizar este lenguaje porque dispone de una librería muy sencilla para trabajar con RabbitMQ.

Sitio web: https://www.python.org/

4.2 Infraestructura Web

Podemos clasificar el stack tecnológico elegido para el desarrollo de la infraestructura web en tres categorías, la plataforma cloud, arquitectura cloud, herramientas de desarrollo. A continuación se describe cada una de las tecnologías utilizadas.

AWS Cloud

Figura 4-8. Logo de AWS [21]

AWS es el nombre de la plataforma de computación en la nube de amazon. Básicamente es una plataforma cloud que ofrece distintos servicios de distinta clase, destinados al desarrollo de software. Estos servicios se clasifican en categorías como cómputo, almacenamiento y blockchain entre otros. Desarrollar este prototipo utilizando una plataforma cloud nos permite reducir al máximo no solo los costes sino también el tiempo de desarrollo.

Justificación de elección: Se eligió esta plataforma de computación en la nube y no otra debido a que es la más pionera, la más usada y la que más servicios ofrece de todas las del mercado. Además, esta plataforma es la más avanzada en la tecnología serverless y sin duda la que más comunidad tiene.

Sitio web: https://aws.amazon.com/es/

Servicios AWS

Como se dijo anteriormente, dentro de AWS Cloud podemos encontrar distintos servicios. En la misma plataforma existen servicios destinados a una misma funcionalidad, por ello es importante recalcar la razón de uso de cada uno de ellos. Con el fin de explicar cada servicio utilizado para el desarrollo del prototipo y sus correspondientes justificaciones se ha creado una tabla explicativa de los servicios, esta tabla informativa se encuentra en la sección de anexos bajo el nombre de *Tabla de servicios de AWS*.

Serverless

Serverless es un método de computación en la nube basado en el modelo bajo demanda. Siguiendo una arquitectura serverless tu aplicación no requerirá tener su propio servidor sino que hará uso de servidores de alguna plataforma cloud. A medida que lo necesite, tu aplicación reservará tiempo de cómputo en un servidor de tu proveedor cloud, para ello acaparará recursos que liberará una vez termine cada proceso, ahorrando costes.

Justificación de elección: Se eligió esta opción porque dada la naturaleza de la aplicación, el número de procesos a ejecutar por los servidores no es constante y por ende alquilar un servidor supondría un gasto innecesario y en ocasiones un cuello de botella. Usar serverless nos permite escalar automáticamente, reducir costes y facilitar el mantenimiento.

Serverless Framework

Figura 4-9. Logo de Serverless Framework [22]

Es el principal framework de desarrollo para aplicaciones serverless. Nos permite trabajar con una gran variedad de proveedores cloud, entre los cuales se encuentra AWS.

Justificación de elección: Se eligió esta opción dado que es el líder del mercado, el que más usuarios y documentación tiene. Además aunque existen otras opciones como podría ser CloudFormation o Terraform, Serverless Framework es la única herramienta de desarrollo plenamente orientada a arquitecturas serverless.

Sitio web: https://www.serverless.com/

JWT

Para la autenticación de usuarios en la aplicación hemos utilizado la tecnología JWT, también definida como Json Web Token. Se trata de una tecnología de autenticación, basada en tokens donde los tokens, no solo tienen la misión de autenticar sino que contienen información acerca del usuario.

Justificación de elección: Se eligió esta tecnología porque tiene fácil integración con el servicio Amazon Cognito de AWS.

Sitio web: https://jwt.io/

Node.js

Figura 4-10. Logo de Node.js [23]

Este es el lenguaje de programación que se escogió para el desarrollo de la infraestructura web. Node.js es el lenguaje de desarrollo serverless por excelencia.

Justificación de elección: Se eligió este lenguaje por varias razones, entre las que se encuentra el hecho de ser el más utilizado, el que más librerías tienes y mejor compatibilidad ofrece con los servicios de AWS y con Serverless Framework.

Sitio web: https://nodejs.org/en/

4.3 Aplicación móvil

En esta sección tratamos las tecnologías utilizadas para el desarrollo de la aplicación móvil, las cuales son, el lenguaje de programación y el framework utilizados.

Flutter

Figura 4-11. Logo de Flutter [24]

Flutter es un UI toolkit de Google creado para desarrollar aplicaciones bonitas, nativamente para móvil, web y escritorio con un mismo código fuente. Nos permite desarrollar a partir de un mismo código aplicaciones que se ejecutan en casi cualquier plataforma, se basa en un sistema de widgets que nos permite reutilizar elementos creados por otros usuarios.

Justificación de elección: Se eligió utilizar esta tecnología por su versatilidad, ya que se ejecuta en cualquier plataforma y la eficiencia que le proporciona ser nativo.

Sitio web: https://flutter.dev/web

Dart

Figura 4-12. Logo de Flutter [25]

Dart es un lenguaje optimizado para clientes, cuyo objetivo es permitir la ejecución de aplicaciones de forma rápida en cualquier plataforma. Es el lenguaje en el que se ha desarrollado el toolkit Flutter.

Justificación de elección: Se utiliza este lenguaje porque es un requisito para poder utilizar Flutter.

Sitio web: https://dart.dev/

4.4 Herramientas generales

En este apartado se tratan las herramientas y tecnologías generales utilizadas a lo largo del desarrollo del proyecto.

Github

Figura 4-13. Logo de GitHub [26]

Es una plataforma online, que ofrece un software para almacenamiento de software y control de versiones.

Justificación de elección: Se utiliza esta plataforma porque es la más extendida.

Sitio web: https://github.com/

CloudCraft

Figura 4-14. Logo de CloudCraft [27]

CloudCraft es un software de computación en la nube, que sirve para diseñar, administrar e implementar servicios de infraestructura en la nube. En esta ocasión se utilizó para crear imágenes de arquitectura de servicios en AWS.

Justificación de elección: Se utiliza esta plataforma por la experiencia previa existente.

Sitio web: https://www.cloudcraft.co/

Draw.io

Figura 4-15. Logo de Draw.io [28]

Herramienta enfocada en el diseño de gráficos relacionados con el mundo del software. Fue utilizado para crear imágenes del diseño de la arquitectura.

Justificación de elección: Se utiliza esta plataforma porque es la más extendida.

Sitio web: https://drawio-app.com/

Adobe Xd

Figura 4-16. Logo de Adobe Xd [29]

Herramienta de creación de interfaces de usuario. Esta herramienta se usa para la creación de los Mockups de la aplicación móvil.

Justificación de elección: Se utiliza esta plataforma por la experiencia previa existente.

Sitio web: https://www.adobe.com/products/xd.html

Visual Studio Code

Figura 4-17. Logo de Visual Studio Code [30]

Esta herramienta es un editor de código. Utilizamos esta herramienta para el desarrollo del software de los productores.

Justificación de elección: Se eligió esta herramienta por la alta cantidad de plugins existente para el uso de tecnologías docker.

Sitio web: https://code.visualstudio.com/

WebStorm IDE

Figura 4-18. Logo de WebStorm [31]

WebStorm una IDE orientada al desarrollo con JavaScript. Fue utilizada para el desarrollo de la infraestructura web.

Justificación de elección: Se eligió esta herramienta porque ofrece una gran variedad de facilidades para el desarrollo de software.

Sitio web: https://www.jetbrains.com/webstorm/

Android Studio

Figura 4-19. Logo de WebStorm [32]

Es la IDE oficial de google para el desarrollo de aplicaciones Flutter. Está desarrollada por Jetbrains y google.

Justificación de elección: Se eligió esta herramienta por ser la recomendación de google (Empresa creadora de Flutter).

Sitio web: https://developer.android.com/studio

5. Implementación y desarrollo

Como parte de este trabajo, se lleva a cabo la implementación de un prototipo de la aplicación planteada en los apartados de Análisis y Diseño. En este apartado se tratará de abordar el proceso de desarrollo de cada una de las piezas desarrolladas. Con el objetivo de exponer el proceso de la forma más sencilla posible, se pueden encontrar distintas secciones que se corresponden con las diferentes fases del desarrollo.

Los apartados definidos son los siguientes:

- Infraestructura o Backend
- Cámara
- Aplicación móvil

5.1 Infraestructura o Backend

Esta fase de desarrollo se corresponde con la implementación de la infraestructura web necesaria para dar soporte a la aplicación.

A lo largo del desarrollo de la misma se llevaron a cabo varias iteraciones, lo cual dio lugar a la aparición y desaparición de distintos microservicios. A continuación se definen las distintas iteraciones y sus correspondientes requisitos, resultados y evaluaciones.

5.1.1 Primera etapa de desarrollo

Requisitos

Como objetivo de esta primera etapa, se pretendía comprobar el funcionamiento de los distintos servicios de la plataforma *Amazon Web Services* necesarios para desarrollar la infraestructura. Por ello se establecieron los siguientes requisitos:

- Crear un servicio capaz de indexar y analizar caras.
- Crear un servicio capaz de proveernos de autenticación.

Resultados

Resultado de esta primera etapa nacieron dos servicios plenamente funcionales. Estos dos servicios resultantes eran *Face Recognition Service* y *Authentication Service*.

El primer servicio obtenido era capaz de indexar y analizar caras. Este servicio, denominado Face Recognition Service, proporcionó toda la infraestructura necesaria para realizar el reconocimiento facial.

El segundo servicio obtenido, denominado *Authentication Service,* proporcionó los recursos necesarios para tener un sistema de autenticación a través de una API propia. Este segundo servicio estaba basado en la tecnología Congnito de *Amazon Web Services*.

Fruto de la evaluación de esta etapa se obtuvieron dos conclusiones que marcarían el resto del desarrollo. La primera conclusión fue que el *Authentication Service* era innecesario y la segunda que *Face Recognition Service* debía dividirse en dos servicios independientes.

Tras la evaluación del *Authentication Service* se concluyó que suponía un sobrecoste de mantenimiento innecesario. Esto se debía a que podía permitirse a la propia aplicación conectarse con la plataforma de autenticación sin necesidad de utilizar una API intermedia. Este hecho, sumado a los posibles fallos de seguridad que podría ocasionar tratar de forma directa con las credenciales de los usuarios, produjo que se tomará la decisión de prescindir de este servicio, delegando así en la aplicación la labor de autenticación en la plataforma *Congnito de Amazon Web Service*.

En cuanto al *Face Recognition Service* se observó la existencia de dos conjuntos de funcionalidades independientes. Un primer conjunto asociado a la indexación de caras y un segundo conjunto asociado al análisis de imágenes. Fruto de este análisis se concluyó que se debía dividir el servicio desarrollado en dos.

5.1.2 Segunda etapa de desarrollo

Requisitos

Los objetivos de esta segunda etapa se encuentran dirigidos a corregir los errores observados en la etapa previa fruto de su evaluación. De acuerdo a estos objetivos se definen los siguientes requisitos:

- Separar *Face Recognition Service* en dos servicios independientes.
- Prescindir del servicio de *Authentication Service* delegando la responsabilidad en la aplicación móvil.

Resultados

Tras el desarrollo de esta segunda etapa se obtuvo dos servicios. El primer servicio obtenido estaba asociado a la indexación de caras y se le denominó *Indexer Service*. El segundo servicio obtenido estaba asociado al análisis facial y se denominó *Analyzer Service*.

El primer servicio obtenido, *Indexer Service,* nos brindaba toda la infraestructura necesaria para llevar a cabo la indexación de nuevas caras en nuestra base de datos.

El segundo servicio obtenido, *Analyzer Service,* nos brindaba toda la infraestructura necesaria para autenticar caras en base a nuestra base de datos.

Tras la evaluación de los servicios producto de esta etapa de desarrollo, se concluyó que habían cumplido de forma satisfactoria con los requisitos planteados.

5.1.3 Tercera etapa de desarrollo

Requisitos

Los objetivos de esta tercera etapa se corresponden con la creación de un nuevo servicio. Este nuevo servicio pretende proporcionar la infraestructura necesaria para registrar eventos capturados por las cámaras. De acuerdo con el objetivo planteado se establece un único requisito:

- Crear un servicio que nos permita registrar los eventos capturados por las cámaras.

Resultados

Como resultado de esta etapa se obtuvo un nuevo servicio denominado *Event Service*. Este nuevo servicio es capaz de registrar todos los eventos capturados por las cámaras para así permitir su posterior utilización por parte del resto de servicios.

El resultado de la evaluación de esta etapa fue positiva, dando así por finalizado el desarrollo del servicio creado. Tras el desarrollo del *Event service* se consideró necesario desarrollar un nuevo servicio que permitiese controlar las cámaras, esto supone dar al usuario la opción de apagado y encendido de los productores a través de la aplicación móvil.

5.1.4 Cuarta etapa de desarrollo

Requisitos

En esta cuarta etapa se pretende cubrir una necesidad surgida en la etapa de evaluación de la etapa anterior. Para ello se propone crear un servicio que permita el apagado y encendido de las distintas cámaras desde la aplicación móvil. De acuerdo a lo anteriormente expuesto se define el siguiente requisito:

- Desarrollar un servicio que permita al usuario dejar de registrar eventos producidos por una cámara y a su vez reanudar el registro de dichos eventos.

Resultados

El desarrollo de esta etapa supuso la creación de un nuevo servicio definido como *Controller Service*, este servicio nos proporcionó la capacidad de, por un lado, dejar de registrar eventos capturados por una cámara y por otro reanudar el registro de eventos de una cámara. El desarrollo de esta etapa fue exitoso y dio lugar a un servicio que recogía los requisitos planteados. Con la finalización de esta etapa finalizó el desarrollo de la Infraestructura.

5.2 Productor o Cámara

Esta fase de desarrollo se corresponde con la construcción de una cámara que captura imágenes cuando detecta movimiento y que además es capaz de utilizar el servicio *Event Service* desarrollado en la fase anterior para registrar los eventos.

El desarrollo de la cámara tiene dos componentes claramente diferenciados, el primer componente es el software y el segundo componente el hardware.

A continuación abordaremos el desarrollo de los distintos componentes en orden de implementación.

5.2.1 Primera etapa

Requisitos

Esta etapa se caracteriza por el desarrollo de un programa que nos permite capturar movimiento a través de la cámara. De acuerdo con esta definición se establecen los siguientes requisitos:

- Crear un programa capaz de detectar movimiento utilizando la cámara
- Permitir ejecutar este programa en el mayor número de dispositivos

Resultados

De acuerdo con el primer requisito se creó un programa capaz de capturar imágenes al detectar movimientos con la cámara. En cuanto al segundo, supuso la creación de un contenedor, esto permitió ejecutar el programa en cualquier dispositivo con un sistema operativo compatible con Docker sin necesidad de personalizar nada.

Como resultado de esta etapa se obtuvo un programa que se ejecutaba dentro de un contenedor. Este hecho supone que se pueda ejecutar en cualquier entorno. En base a este resultado se decidió utilizar un modelo de microservicios para facilitar así la ejecución del programa.

5.2.2 Segunda etapa

Requisitos

Esta etapa pretende dar la capacidad a las distintas cámaras de registrar las capturas que han realizado. Con el fin de cumplir lo anteriormente expuesto se plantean los siguientes requisitos:

- Aportar al programa anteriormente creado la capacidad de subir automáticamente las imágenes capturadas por la cámara a nuestra base de datos de imágenes.

Resultados

Resultado de esta etapa se obtuvo un software capaz de capturar imágenes y subirlas a nuestra base de datos en la nube. Este programa además es instalable en cualquier

plataforma Unix. La implementación de esta etapa terminó con el desarrollo software dando paso al desarrollo del hardware.

5.2.3 Tercera etapa

Requisitos

En esta etapa se creó un dispositivo hardware con todos los componentes necesarios para ejecutar el software creado. Se plantean para esta etapa los siguientes requisitos:

- El dispositivo debe tener una cámara
- El dispositivo debe de tener un sistema operativo compatible con Docker
- El dispositivo debe tener suficiente memoria como para almacenar una gran cantidad de imágenes
- El dispositivo debe tener conexión a internet vía wifi
- El dispositivo debe de ser transportable
- El dispositivo debe de tener un coste bajo

Resultado

Como resultado de esta etapa se obtuvo un dispositivo conformado por una Raspberry 4 con una cámara y un sistema operativo Unix desde el cual se puede ejecutar el software desarrollado. Con esta etapa se finalizó con el desarrollo de la cámara para dar paso al desarrollo de la app.

5.3 Aplicación móvil

Esta fase de desarrollo se corresponde con la creación de una aplicación móvil capaz de interactuar con la infraestructura creada.

Esta fase se encuentra dividida en las siguientes tres etapas:

1. Creación de clientes y servicios
2. Creación del modelo y adaptación de los servicios
3. Creación de la Interfaz de usuario

Este orden es necesario dado que el desarrollo previo de los servicios, los clientes y el modelo, nos habilita a desarrollar la interfaz de usuario.

5.3.1 Creación de clientes y servicios

Requisitos

En esta etapa el objetivo es crear todos los clientes y servicios necesarios para el desarrollo de la aplicación. Los servicios necesarios son:

- Servicio de autenticación
- Servicio del controlador
- Servicio de eventos
- Servicio de indexado
- Servicio de almacenamiento

Los clientes necesarios son:

- Cliente de autenticación
- Cliente del controlador
- Cliente de eventos
- Cliente de indexado

Resultado

Como resultado de esta etapa se obtienen todos los servicios y clientes planteados en la sección de requisitos.

5.3.2 Creación del modelo y adaptación de los servicios

Requisitos

En esta etapa se enfoca en la creación de las clases necesarias para definir los modelos de datos de la aplicación y adaptar nuestros servicios a estos modelos. Los modelos a desarrollar son:

- Event
- EventList
- Producer
- Record

Cada uno de estos modelos se corresponde con uno tipo de dato obtenido a partir del procesamiento de las respuestas de los servicios web creados.

Resultado

Como resultado de esta etapa se obtienen todos los modelos planteados en la sección de requisitos. Además, se hace necesario llevar a cabo una modificación en los servicios creados previamente, con el fin de adaptarlos a estos modelos.

5.3.3 Creación de la Interfaz de usuario

Requisitos

En esta etapa el objetivo es crear todos los widgets (elementos visuales de la aplicación) necesarios para crear la aplicación móvil. Esta etapa es la más creativa y se compone de la creación de las distintas vistas de la aplicación, incluyendo cada uno de los elementos visuales presentes.

Los elementos creados en esta etapa se complementan con los creados en las previas para así obtener la información a representar.

Resultado

Esta etapa se llevó a cabo satisfactoriamente y como resultado se obtuvo la aplicación finalizada.

6. Conclusión

Esta aplicación tenía como objetivo validar la viabilidad de un sistema de videovigilancia con reconocimiento facial basado en un modelo serverless. Para llegar a una conclusión acerca del resultado obtenido se evalúan diferentes puntos.

El primer punto a evaluar es la capacidad tecnológica. Las arquitecturas serverless son relativamente jóvenes, por este motivo todavía quedan muchos usos por estudiar. Crear un sistema de estas características desde cero, utilizando exclusivamente este tipo de arquitectura supone un reto a nivel de diseño. Tras la implementación del prototipo, podemos decir que desde el punto de vista tecnológico ha sido un éxito y que efectivamente es posible el desarrollo de un sistema de videovigilancia basado en un modelo serverless exclusivamente.

El segundo punto a tener en cuenta para determinar la viabilidad del prototipo es el coste de ejecución. Como hemos podido ver a lo largo de este documento, resulta extremadamente complicado calcular los costes de una aplicación serverless con exactitud. Esto se debe a que al ser un modelo bajo demanda, los costes son completamente variables. Por otro lado Amazon Rekognition es un servicio relativamente caro, es por este motivo que se plantea la funcionalidad de encendido y apagado del sistema. Si un usuario dejase continuamente el sistema encendido y la cámara detectase movimiento con mucha frecuencia, el gasto asociado a este usuario podría llegar a ser muy significativo. Sin embargo sí se hiciera un uso responsable o se desactivase el reconocimiento facial, sin duda sería un producto mucho más barato que la competencia. Como solución a este punto, se propone crear una funcionalidad que impida al usuario activar el sistema cuando esté en el lugar donde se ha instalado o desactivar el reconocimiento facial permanentemente.

Como resultado de la reflexión anterior podemos decir que efectivamente el uso de serverless abarata los costes del sistema, pero para garantizar que esto sea así, se debe impedir que el usuario se encuentre en el interior del espacio protegido mientras el sistema está activo.

El tercer y último punto a tener en cuenta es la efectividad del reconocimiento facial. El correcto funcionamiento del sistema de reconocimiento facial depende de dos factores principalmente. Estos factores son la resolución de las imágenes tomadas y la orientación de la cámara con respecto al punto de acceso a la zona vigilada. Los costes de obtención

de una cámara capaz de obtener imágenes con la resolución necesaria para realizar un reconocimiento facial efectivo son bajos, por esto se da por validado el primero de los factores. Por otro lado el cumplimiento del segundo factor recae en su totalidad en el usuario final.

Tras la realización de pruebas asociadas al correcto funcionamiento del reconocimiento facial, se concluye que efectivamente es viable la inclusión de esta funcionalidad en el sistema. Sin embargo es importante tener en cuenta que representa la mayor parte del coste del sistema.

En conclusión podemos decir que el resultado del análisis ha sido exitoso, exponiendo la viabilidad de un sistema de videovigilancia con reconocimiento facial basado en un modelo serverless.

7. Anexos

7.1 Ficha técnica Raspberry PI 4

Ficha técnica Raspberry PI 4	
Marca	Raspberry Pi Spain
Fabricante	Raspberry Pi Spain
Modelo	RAS-4-4G
Dimensiones del producto	10 x 7 x 3 cm; 50 gramos
Número de modelo del producto	RAS-4-4G
Número de producto	SC15185
Capacidad de la memoria RAM	4096 MB
Capacidad de la memoria	4 GB
Ranuras de memoria disponibles	1
Capacidad de la memoria RAM instalada	4 GB
Tecnología de la memoria RAM	SDRAM
Tipo de memoria del ordenador	DDR3 SDRAM
Interfaz del disco duro	USB 2.0
Sistema operativo	Windows 10
Velocidad del procesador	1.5
Tipo de procesador	Cortex
Número de procesadores	4
Toma del procesador	LGA 1150
Interfaz del hardware	MicroSD, Bluetooth, Vídeo compuesto, USB, Micro-HDMI, HDMI, USB 2.0
Descripción de la tarjeta gráfica	Integrated

Ficha técnica Raspberry PI 4	
Tipo de memoria gráfica	DDR3 SDRAM
Interfaz de la tarjeta gráfica	AGP
Componentes incluidos	Placa
Número de productos	1
Compilador	Raspberry-Pi
Pantalla a color	No
Pilas / baterías incluidas	No
Pilas / baterías necesarias	No
Adaptador	Raspberry-Pi
Tipo de conexión inalámbrica	Bluetooth
Número de puertos USB	4
Lector	Raspberry-Pi
Entrada de interfaz humana	Teclado
Marimba	Raspberry-Pi
Grabador	Raspberry-Pi
Transcriptor	Raspberry-Pi
Enfoque automático	No
Programable	No
Peso del producto	50 g

Tabla 7-1. Ficha técnica de RaspBerry Pi 4 [33]

7.2 Ficha técnica cámara

Ficha técnica cámara	
Marca	Longruner
Fabricante	Longruner
Modelo	LC26-US
Dimensiones del paquete	11.6 x 5.2 x 2.8 cm; 20 gramos
Número de modelo del producto	LC26-US
Número de producto	LC26-US
Pilas / baterías incluidas	No
Pilas / baterías necesarias	No
Tipo de lente	Gran angular
Enfoque automático	No
Programable	No
Peso del producto	20 g

Tabla 7-2. Ficha técnica de Cámara [35]

7.3 Tabla de servicios de Amazon Web Services utilizados

Icono	Nombre	Descripción
	**DynamoDB**	DynamoDB es un servicio de base de datos noSQL de Amazon Web Services. Una de sus principales características es que es una base de datos distribuida. Utilizamos este servicio como base de datos para todas las tablas creadas.
	**S3**	S3 es un servicio de almacenamiento de objetos de Amazon Web Services, que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento. Utilizamos este servicio para almacenamiento de imágenes y de datos en formato JSON.
	**Amazon Rekognition**	Es un servicio de reconocimiento facial bajo demanda. Este servicio nos permite crear diccionarios de caras y analizar imágenes. Utilizamos este servicio para saber si en las imágenes analizadas se encuentra o no una cara indexada indexada por el usuario en el sistema.
	**Lambda**	AWS Lambda es un servicio de cómputo bajo demanda de Amazon Web Servicies. Con este servicio en lugar de pagar un servidor, se paga por el tiempo de ejecución utilizado. Utilizamos este servicio como servicio de cómputo para nuestra infraestructura cloud.
	**API Gateway**	Es un servicio de Amazon Web Servicies que facilita a los desarrolladores la administración, creación, publicación y monitoreo de API a cualquier escala. Utilizamos este servicio para crear las APIs desarrolladas como parte de nuestra infraestructura web.

Icono	Nombre	Descripción
	**Amazon Cognito**	Es un servicio de Amazon Web Services que facilita la creación y administración de pools de usuarios. Utilizamos este servicio para la gestión de usuarios de nuestro sistema.
	**Cloud Formation**	Es un servicio de Amazon Web Services que nos permite desplegar infraestructura a través de una plantilla. Este servicio es utilizado por Serverless Framework, para el despliegue de la infraestructura.

Tabla 7-3. Tabla de servicios AWS [36]

7.4 Análisis de coste de servicios de AWS

En esta sección se estudian los costes de los principales servicios utilizados. Algunos de los costes mostrados a continuación son el resultado de una simplificación de los costes reales. Esto es así debido a que cada uno de los servicios tiene una política de costes única y compleja, cuyo estudio se escapa del objetivo de este proyecto. A la hora de analizar los precios hay que tener en cuenta que AWS ofrece distintas regiones con distintos precios. Todos los precios de este documento son relativos a la región de Irlanda, aquella en la que se ha desplegado nuestra arquitectura.

Todos los costes mostrados están referidos al momento de elaboración de este documento (Noviembre de 2020).

Amazon Rekognition

Amazon Rekognition nos ofrece dos servicios, cada uno con su coste asociado, estos servicios son *Amazon Rekognition Image* y *Amazon Rekognition Video*. En este caso solo analizaremos el coste asociado a *Amazon Rekognition Image* dado que es el utilizado en este trabajo.

En *Amazon Rekognition Image* el coste se encuentra dividido en dos categorías, la primera la definen los costes del procesamiento de imagen y la segunda los costes de almacenamiento de los metadatos de las caras indexadas.

Costes de procesamiento

El precio de procesamiento de imágenes varía en función del número de imágenes procesadas. A continuación se muestra la tabla de costes de procesamiento de imágenes.

Capa	Coste por imagen procesada
Primer millón de imágenes procesadas del mes	0,00085 €
Siguiente 9 millones de imágenes procesadas del mes	0,00068 €
Siguiente 90 millones de imágenes procesadas del mes	0,0005 €
Siguiente 100 millones de imágenes procesadas del mes	0,0003 €

Tabla 7-4. Tabla de costes de procesamiento de imágenes con Amazon Rekognition [36]

Costes almacenamiento de los metadatos

Los costes asociados al almacenamiento de los metadatos de las caras indexadas, al requerir un acaparamiento continuo de recursos, son costes recurrentes. El coste de almacenamiento por cara es de 0,0000085€ por mes.

Lambda

Hay dos variables determinantes para el cálculo del coste de procesamiento con *Lambda*. Estas variables son la memoria RAM asignada y el tiempo de cómputo. En esta ocasión todas las Lambdas creadas tienen asignada memoria de 128MB. Por esta razón, a partir de este punto todos los costes provistos estarán asociados a *Lambda* de 128MB de memoria.

El coste de ejecución de una *Lambda* varía en función del número de ejecuciones realizadas sobre este servicio. De esta forma nos encontramos con la tabla 7-5:

Capa	Coste por imagen procesada
Primer millón ejecuciones del mes	0 €
Resto de ejecuciones	0,00000018 € por cada 100 ms de ejecución

Tabla 7-5. Tabla de costes de ejecución de Lambda [36]

API Gateway

El coste de este servicio viene determinado por el número de peticiones http realizadas. A continuación se muestra el coste asociado al mismo.

Capa	Coste por imagen procesada
Primer millón de peticiones del mes	0 €
Hasta 300 millones de peticiones en el mes	0,00000085 €
Resto de peticiones	0,00000076 €

Tabla 7-6. Tabla de costes de API GateWay [36]

DynamoDB

DynamoDB tiene dos modos de capacidad, estos son el modo de capacidad bajo demanda y el modo de capacidad aprovisionada. En este caso hacemos uso del modo de capacidad bajo demanda. Los costes de este modo se dividen en dos tipos de gasto.

El primer tipo de gasto al que nos atenemos es el asociado al almacenamiento. Los costes de asociados al almacenamiento están definidos por los siguientes puntos:

- Los primeros 25 GB almacenados cada mes son gratis
- 0,24 € por GB mes de ahí en adelante

El segundo tipo de gasto al que nos atenemos es el referido a la transferencias de datos. En las siguientes tabla podemos ver estos costes.

Tipo de transferencia de datos	Coste por transferencia
Escritura	0,0000012 €
Lectura	0,00000024 €

Tabla 7-7. Tabla de costes de DynamoDB [36]

S3

El servicio de *S3* tiene varias opciones de uso. Nosotros utilizamos *S3* standard cuyos costes se muestran en la siguiente tabla:

Almacenamiento	Coste
Primeros 50 TB/mes	0,02 € por GB
Siguientes 450 TB/mes	0,019 € por GB
El resto de 500 TB/mes	0,018 € por GB

Tabla 7-8. Tabla de costes de S3 [36]

Amazon Cognito

El coste de este servicio lo define el número de usuarios mensualmente activo (MAU). El servicio ofrece gratuitamente hasta 50.000 usuarios. Y pasado ese número de usuarios nos encontramos los precios mostrados en la siguiente tabla.

Transferencia de datos	Coste
Después de la capa gratuita de 50.000	0,0047 €
Siguientes 900.000	0,0039 €
Siguientes 9.000.000	0,0028 €
Más de 10.000.000	0,0021 €

Tabla 7-9. Tabla de costes de Amazon Cognito [36]

7.5 Enlace a repositorios de trabajo

En esta sección listamos todos los repositorios creados. Todos los repositorios se encuentran en https://github.com/.

Productor

El software asociado al productor se compone de dos repositorios.

Software de detección de movimientos

- https://github.com/alerodang/ArgusCameraMotionDetection

Software de productor

- https://github.com/alerodang/ArgusCameraModule

Infraestructura web

Por cada uno de los servicios creados como parte de la infraestructura web se creó un repositorio.

Indexer Service

- https://github.com/alerodang/ArgusIndexerService

Analyzer Service

- https://github.com/alerodang/ArgusAnalyzerService

Event Service

- https://github.com/alerodang/ArgusEventService

Controller Service

- https://github.com/alerodang/ArgusControllerService

Infraestructura móvil

La aplicación móvil se encuentra en el siguiente repositorio:

https://github.com/alerodang/ArgusApp

7.6 Manual de usuario

En esta sección se explica como hacer uso de las principales funcionalidades de la aplicación. Encontramos dos secciones, la primera referente a la aplicación móvil y la segunda a la creación de productores. En estos manuales se asume desplegada la infraestructura web de la aplicación. Toda la información acerca del despliegue de la infraestructura web se encuentra en los archivos README.md de los correspondientes repositorios.

7.6.1 Aplicación móvil

En todas estas funcionalidades se asume que el usuario se ha identificado en la aplicación.

Visualización de accesos registrados

Para visualizar los accesos registrados, el usuario debe dirigirse a la vista de accesos registrados. Esta vista nos muestra el listado de accesos registrados por el sistema, conjuntamente con información sobre estos. En el caso de no haberse registrado ningún acceso, se muestra un mensaje.

En los elementos de esta lista podemos apreciar información sobre el productor que registró el acceso, la fecha del acceso y es un acceso permitido o no.

Visualización de detalles de accesos registrados

Para obtener más información sobre un acceso registrado el usuario debe pulsar sobre la tarjeta asociada a dicho acceso en la vista de accesos registrados. Esta tarjeta se volteara y le mostrará a la persona que se ha identificado o en su defecto un mensaje que indica que es un acceso no permitido.

Visualización de imágenes de accesos registrados

Para ver las imágenes asociadas a un acceso registrado, el usuario debe ir a la vista de accesos registrados, voltear el acceso cuyas imágenes quiere visualizar y picar en el icono de imágenes en la parte inferior derecha de la tarjeta. Tras esto se le conducirá a una nueva vista en la que podrá ver las distintas capturas tomadas durante el acceso. En esta nueva vista se presenta un carrusel que contiene todas las imágenes tomadas, el usuario puede deslizar las imágenes para avanzar entre estas.

Indexar nueva cara

Para indexar una nueva cara el usuario tiene que dirigirse a la vista de indexación de caras. En esta vista el usuario debe pulsar en el icono de la cámara y sacar una foto frontal de la cara de la persona a registrar en el sistema. Acto seguido deberá introducir una foto frontal-lateral-izquierda y otra frontal-lateral-derecha. Tras añadir las fotos, el usuario debe introducir el nombre de la persona registrada en el formulario que se le muestra en la vista.

Una vez introducidos todos los datos se habilitará el botón de envío. Cuando el usuario pulse este botón se llevará a cabo el registro.

Registrar nuevo productor

Para registrar un nuevo productor, el usuario debe dirigirse a la sección de productores a través de la barra de navegación de la aplicación. En esta vista deberá pulsar el botón de añadir nuevo productor, el cual se corresponde con un símbolo de suma dentro de un círculo. Al pulsar este botón se le muestra un formulario a rellenar, en este formulario debe introducir el nombre del productor y el secreto de este para darle de alta en el sistema.

Una vez introducidos los datos el usuario debe pulsar en el botón de envío del formulario para registrar el producto.

Apagar y encender productor

Para apagar o encender un productor, el usuario debe dirigirse a la sección de productores a través de la barra de navegación de la aplicación. En esta vista deberá localizar el productor cuyo estado quiere modificar. A cada productor se le asocia un botón de encendido y apagado, el estado del productor lo define el color de este botón.

Sí queremos apagar un productor, debemos asegurarnos de que el botón en cuestión este de color azul y pulsarlo. Para encenderlo, debemos asegurarnos de que este en rojo y pulsarlo. Ambas acciones modifican el color del botón apagando y encendiendo el productor respectivamente.

7.6.2 Creación de productor

Para crear un productor, debemos hacernos con el siguiente hardware:

- Computador con un sistema operativo Unix instalado, compatible con docker.

- Cámara compatible con nuestro hardware.

Una vez dispongamos del hardware necesario debemos:

1. Instalar docker y docker-compose en el equipo.
2. Conectar cámara al equipo (solo si no está integrada).
3. Descargar el software de productor (especificado en el punto 7.5)
4. Seguir las instrucciones del README del repositorio.

Hay que tener en cuenta que el nombre del productor y del secreto del productor introducidos en el productor y en la aplicación deben coincidir.

8. Glosario de términos

Framework: Un entorno de trabajo (del inglés framework), o marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. En el desarrollo de software, un entorno de trabajo es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software [1].

Backend y frontend: Front end y back end son términos que se refieren a la separación de intereses entre una capa de presentación y una capa de acceso a datos, respectivamente. Según el contexto, para referirse a front end se usan otros términos como frontal o interfaz de usuario,1 mientras que a back end se le llama servidor, motor o modo administrador [2].

Aplicación nativa: La aplicación nativa está desarrollada y optimizada específicamente para el sistema operativo determinado y la plataforma de desarrollo del fabricante (Android, iOS, etc). Este tipo de aplicaciones se adapta al 100% con las funcionalidades y características del dispositivo obteniendo así una mejor experiencia de uso [3].

Aplicación multiplataforma: Una aplicación multiplataforma, es una aplicación capaz de correr en más de un sistema operativo.

API: Una API es una interfaz de programación de aplicaciones (del inglés API: Application Programming Interface). Es un conjunto de rutinas que provee acceso a funciones de un determinado software [4].

Arquitectura basada en microservicios: La arquitectura de microservicios (en inglés, Micro Services Architecture, MSA) es una aproximación para el desarrollo de software que consiste en construir una aplicación como un conjunto de pequeños servicios, los cuales se ejecutan en su propio proceso y se comunican con mecanismos ligeros (normalmente una API de recursos HTTP). Cada servicio se encarga de implementar una funcionalidad completa del negocio. Cada servicio es desplegado de forma independiente y puede estar programado en distintos lenguajes y usar diferentes tecnologías de almacenamiento de datos [5].

Cliente: Es una aplicación informática o un ordenador que consume un servicio remoto en otro ordenador conocido como servidor, normalmente a través de una red de

telecomunicaciones. También se puede definir como cualquier pieza software que se conecta a un servidor [6].

Productor: Un productor en el contexto de este trabajo, es un dispositivo hardware que recoge información del entorno y la transmite al sistema. Un ejemplo de producto es la cámara creada en el prototipo.

Secreto del productor: El secreto del productor es la clave que se crea el para sincronizar el productor con el sistema.

Colección: Una colección es un conjunto de caras asociada a una cuenta. Un usuario puede añadir caras a las colecciones en cualquier momento haciendo uso del Indexer Service.

9. Referencias

1. Colaboradores de Wikipedia «Framework». Disponible: https://es.wikipedia.org/wiki/Framework. [Último acceso: 19 Noviembre 2020].

2. Colaboradores de Wikipedia «Front End y Back End». Disponible: https://es.wikipedia.org/wiki/Front_end_y_back_end. [Último acceso: 19 Noviembre 2020].

3. Colaboradores de www.raona.com «¿App nativa, web o híbrida?». Disponible: https://www.raona.com/aplicacion-nativa-web-hibrida/#:~:text=La%20aplicaci%C3%B3n%20nativa%20est%C3%A1%20desarrollada,una%20mejor%20experiencia%20de%20uso. [Último acceso: 19 Noviembre 2020].

4. Colaboradores de Wikipedia «Web Api». Disponible: https://es.wikipedia.org/wiki/Web_API#:~:text=Una%20API%20es%20una%20interfaz,funciones%20de%20un%20determinado%20software. [Último acceso: 19 Noviembre 2020].

5. Colaboradores de Wikipedia «Arquitectura de microservicios». Disponible: https://es.wikipedia.org/wiki/Arquitectura_de_microservicios. [Último acceso: 19 Noviembre 2020].

6. Colaboradores de Wikipedia «Cliente informático». Disponible: https://es.wikipedia.org/wiki/Cliente_(inform%C3%A1tica). [Último acceso: 19 Noviembre 2020].

7. Página de Securitas Direct en Play Store. Disponible: https://play.google.com/store/apps/details?id=com.securitasdirect.android.mycontrol&hl=en&gl=US. [Último acceso: 19 Noviembre 2020].

8. https://www.securitasdirect.es/. [Último acceso: 19 Noviembre 2020].

9. Página de WardenCam360 en Play Store. Disponible: https://play.google.com/store/apps/details?id=com.warden.cam&hl=en_US&gl=US. [Último acceso: 19 Noviembre 2020].

10. Página web de WardenCam360. Disponible: http://www.wardencam360.com/. [Último acceso: 19 Noviembre 2020].

11. Colaboradores de telcasa.net «Protección de datos en el tratamiento de imágenes». Disponible: https://tecalsa.net/proteccion-de-datos-en-el-tratamiento-de-imagenes/. [Último acceso: 19 Noviembre 2020].

12. Colaboradores de Wikipedia «Ley Orgánica de protección de datos». Disponible: https://es.wikipedia.org/wiki/Ley_Org%C3%A1nica_de_Protecci%C3%B3n_de_Datos_de_Car%C3%A1cter_Personal_(Espa%C3%B1a). [Último acceso: 19 Noviembre 2020].

13. Colaboradores de conceptosjuridicos.com «Estatuto de los trabajadores». Disponible: https://www.conceptosjuridicos.com/estatuto-de-los-trabajadores/. [Último acceso: 19 Noviembre 2020].

14. Colaboradores de Wikipedia «Raspberry Pi». Disponible: https://en.wikipedia.org/wiki/Raspberry_Pi. [Último acceso: 19 Noviembre 2020].

15. Colaboradores de Wikipedia «Docker software». Disponible: https://es.wikipedia.org/wiki/Docker_(software). [Último acceso: 19 Noviembre 2020].

16. Repositorio de Docker Compose. Disponible: https://github.com/docker/compose. [Último acceso: 19 Noviembre 2020].

17. Colaboradores de Wikipedia «Lenguaje de programación Java». Disponible: https://en.wikipedia.org/wiki/Java_(programming_language). [Último acceso: 19 Noviembre 2020].

18. Colaboradores de Wikipedia «OpenCV». Disponible: https://en.wikipedia.org/wiki/OpenCV. [Último acceso: 19 Noviembre 2020].

19. Colaboradores de Wikipedia «RabbitMQ». Disponible: https://en.wikipedia.org/wiki/RabbitMQ. [Último acceso: 19 Noviembre 2020].

20. Colaboradores de Wikipedia «Lenguaje de programación Python». Disponible: https://en.wikipedia.org/wiki/Python_(programming_language). [Último acceso: 19 Noviembre 2020].

21. Colaboradores de Wikipedia «Amazon Web Services». Disponible: https://en.wikipedia.org/wiki/Amazon_Web_Services. [Último acceso: 19 Noviembre 2020].

22. Página web de Serverless Framework. Disponible: https://www.serverless.com/. [Último acceso: 19 Noviembre 2020].

23. Node.js. Disponible: https://nodejs.org/en/. [Último acceso: 19 Noviembre 2020].

24. Colaboradores de Wikipedia «Lenguaje de programación Node.js». Disponible: https://en.wikipedia.org/wiki/Flutter_(software). [Último acceso: 19 Noviembre 2020].

25. Colaboradores de Wikipedia «Lenguaje de programación Dart». Disponible: https://en.wikipedia.org/wiki/Dart_(programming_language). [Último acceso: 19 Noviembre 2020].

26. Colaboradores de Wikipedia «Github». Disponible: https://en.wikipedia.org/wiki/GitHub. [Último acceso: 19 Noviembre 2020].

27. CloudCraft. Disponible: https://www.cloudcraft.co/. [Último acceso: 19 Noviembre 2020].

28. Draw.io. Disponible: https://drawio-app.com/. [Último acceso: 19 Noviembre 2020].

29. Colaboradores de Wikipedia «Adobe XD». Disponible: https://en.wikipedia.org/wiki/Adobe_XD. [Último acceso: 19 Noviembre 2020].

30. Colaboradores de Wikipedia «Visual Studio Code». Disponible: https://en.wikipedia.org/wiki/Visual_Studio_Code. [Último acceso: 19 Noviembre 2020].

31. Web Storm. Disponible: https://confluence.jetbrains.com/display/WI/WebStorm+IDE. [Último acceso: 19 Noviembre 2020].

32. Android Studio. Disponible: https://developer.android.com/studio. [Último acceso: 19 Noviembre 2020].

33. Página de Cloudformation en la web de Amazon Web Services. Disponible: https://aws.amazon.com/es/cloudformation/. [Último acceso: 19 Noviembre 2020].

34. Raspberry Pi 4 en la tienda online de Amazon. Disponible: https://www.amazon.es/RASPBERRY-Placa-Modelo-SDRAM-1822096/dp/B07TC2BK1X/ref=sr_1_5. [Último acceso: 19 Noviembre 2020].

35. Cámara en la tienda online de Amazon. Disponible: https://www.amazon.es/gp/product/B07R4JH2ZV/ref=ppx_yo_dt_b_asin_title_o00_s00. [Último acceso: 19 Noviembre 2020].

36. Amazon Web Services. Disponible: https://aws.amazon.com/es/. [Último acceso: 19 Noviembre 2020].