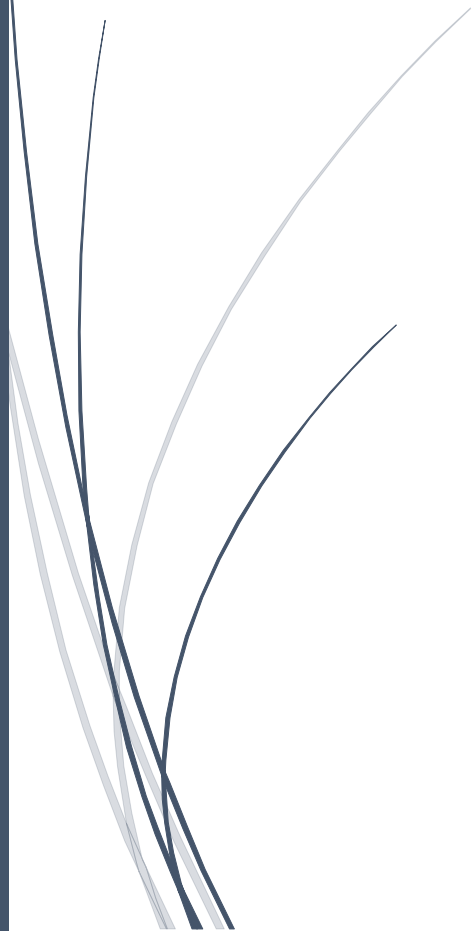
Curso 2019/2020

Sentinel

Aplicación móvil para el control de asistencia en eventos



Miguel Navarro Jorgensen
Tutor Académico: Agustín Rafael Trujillo Pino
Tutor de empresa: Izzat Sabbagh Rodríguez
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Contenido

Glosario de términos

Framework: Puede ser traducido como entorno de trabajo o marco de trabajo. Es una estructura conceptual y tecnológica que definen una serie de normas y módulos software que añaden funcionalidad y sencillez a un producto software.

Microservicio: Es un estilo de arquitectura software que divide las aplicaciones en módulos más pequeños e independientes. Se comunican entre sí para llevar a cabo las mismas tareas de forma conjunta.

Backend: Es un tipo de abstracción que define al conjunto de funciones, rutinas y servicios que son invisibles para el cliente de una aplicación. Ejemplo: El acceso a la base de datos.

Frontend: Es un tipo de abstracción que define la interfaz de usuario un sistema software y con la que este interactúa.

Flutter: Framework desarrollado y propiedad de Google que permite el desarrollo de aplicaciones móviles nativas de forma simultánea para Android y iOS con el lenguaje de programación Dart.

Aplicación nativa: Aplicación móvil desarrollada de forma específica para su ejecución en una plataforma o dispositivo específico.

API: Una API o Application Programming Interface es una interfaz que define las interacciones entre distintos elementos de software. Define el tipo de peticiones que se pueden realizar, de qué forma y con qué datos.

1. Introducción

En la actualidad, la gran mayoría de las personas no salen de casa sin su teléfono móvil. La recepción y envío de información se hace principalmente por canales digitales, aumentando así la productividad de las personas y la capacidad de informarse e informar. Tenemos a nuestro alcance una infinitud de posibilidades gracias a los dispositivos móviles y a la ya tan conocida computación en la nube.

Es por esto por lo que cuando alguien pregunta si alguna idea se puede hacer, mi respuesta suele ser: "Todo lo que quieras se puede hacer, siendo más o menos complejo". Es por esto por lo que resulta curioso ver como en multitud de eventos, ya sea una clase de matemáticas en el colegio o una fiesta privada en la casa de un amigo, el control de las personas que asisten a ese evento se suele realizar de forma manual, ya sea apuntando a los asistentes en un cuaderno de notas, digital o físico, o acordándose de quien ha acudido a la cita y quien no (esta última siendo menos usual). Esta es una tarea muy sencilla de digitalizar, y existen aplicaciones en el mercado que permiten almacenar de una forma relativamente ordenada a los asistentes de un evento. Sin embargo, hay una asignatura pendiente y es la agilización en la forma en la que se registran los asistentes en el evento.

1.1 Objetivos

Los objetivos del proyecto se basan en la aparente necesidad de una aplicación para la gestión completa de la asistencia de personas a diferentes eventos. Es por esto por lo que se pretende construir una aplicación móvil que agilice el registro de nuevos asistentes y la actualización del estado de asistentes previamente registrados en los diferentes eventos.

La aplicación móvil se desarrolla para solventar los problemas de ralentización o atasco que se viven, por ejemplo, en las aulas de la Universidad de Las Palmas de Gran Canaria donde, entre otras fórmulas, el alumno tiene que interrumpir la atención a la clase para apuntarse en una lista, o el profesor tiene que ir preguntando uno por uno a los alumnos para ver quien ha asistido.

Es por esto por lo que, los objetivos del proyecto se describen como la creación de una aplicación móvil que permita al usuario crear y borrar eventos a los cuales se pueden adherir de diferentes modos los asistentes a esos usuarios creados. La información de asistentes y su estado de haber asistido o no al evento, se pueden enviar por email.

Además de los objetivos funcionales, se pretenden cumplir otros objetivos, que son:

- Interfaz amigable a través de la cual se sienta cómoda la interacción del usuario con la aplicación.
- Rendimiento y robustez de la aplicación móvil. Esto incluye un control de errores.
- Coste de mantenimiento de la aplicación. Servicios de la parte "Back end", calidad de código, etc.

1.2 Fases de realización

En este apartado se muestran las diferentes fases sobre las que va a pasar el desarrollo del proyecto, con una breve explicación de en qué consiste cada apartado.

1. **Estudio Previo**: Se realiza una investigación previa a cualquier otro punto del proyecto donde se analiza el estado del arte, las diferentes metodologías para funciones clave en

la aplicación, tecnologías y lenguajes de programación a utilizar para cada apartado del proyecto.

2. **Análisis**: En esta parte del proyecto se analizan los diferentes casos de uso que debe de tener la aplicación, para saber en más detalle qué se va a construir. Es decir, qué funcionalidades tendrá la aplicación.
3. **Diseño**: En este apartado se pretende analizar la arquitectura que tendrá tanto la aplicación móvil como la parte "Backend" de la misma. De esta forma, la posterior implementación y desarrollo del código se hará con unos criterios establecidos.
4. **Implementación y pruebas**: Este apartado es en el cual se desarrollan tanto la aplicación móvil como los servicios que conforman el "Backend" de la aplicación. Este proceso se prueba tras cada implementación.
5. **Validación**: En este apartado se prueba el sistema una vez finalizado con diferentes usuarios desconocedores de la aplicación, para conocer si hay que hacer retoques en la aplicación dada la crítica de estos.

2. Estudio Previo

2.1 Estado del arte

2.1.1 Desarrollo de aplicaciones

En la actualidad, la mayoría de los proyectos de desarrollo de software eligen tecnologías de desarrollo de la aplicación móvil aquellas que permitan un desarrollo simultáneo para los dos sistemas operativos móviles más utilizados: Android y iOS. Las tecnologías para el desarrollo de estas aplicaciones móviles usan un mismo código para generar programas que funcionen con un rendimiento e integraciones similar a como lo hacen aquellas generadas en el lenguaje de programación por defecto de la plataforma. Esto es, Java en los sistemas Android y Swift en iOS.

Al buscar para el desarrollo móvil los frameworks más populares de entre todos, se puede deducir que el más utilizado es Flutter, seguido de React Native, Cordova y Xamarin. [1]

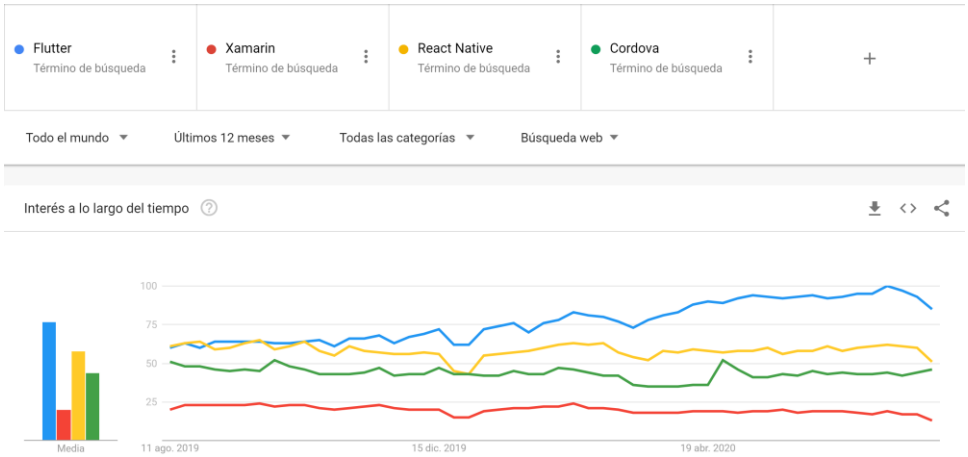


Ilustración 1: Tendencia de búsquedas de frameworks de desarrollo de aplicaciones. Google Trends.

Todos estos frameworks de desarrollo móvil comparten la característica de utilizar un solo código base para funcionar en diferentes sistemas operativos y en un amplio abanico de versiones. Además, estos frameworks al ser populares tienen una amplia comunidad de desarrolladores que preguntan y responden a los problemas o dudas que puedan surgir durante el desarrollo.

Por todo esto (rendimiento, facilidad de desarrollo, mantenibilidad, soporte comunitario, etc.) resulta interesante elegir una de estas tecnologías por encima de una específica de un sistema operativo, dado que Sentinel es una aplicación multiplataforma.

2.1.2 Desarrollo de Backend

Antes la mayoría del software era monolítico. Esto es, un mismo software en un mismo sistema físico daba acceso a la interfaz de usuario, a la capa de datos y al manejo de la lógica del programa. Todas estas partes estaban muy acopladas la una con la otra. Si bien hay algunas aplicaciones críticas, como determinados softwares de finanzas que siguen utilizando este tipo de arquitectura para determinadas partes del negocio, la realidad es que esta arquitectura de software conlleva a unos elevados gastos de mantenimiento del código (entendido como evolución y correcciones del software), de escalabilidad, etc.

El software normalmente no deja de mantenerse y evolucionar en nuevas funcionalidades y en cambiar las ya existentes. Es por esto y por lo explicado anteriormente, que la tendencia de desarrollo de software ha cambiado hacia el modelo basado en microservicios. Esta arquitectura se explica como una colección de múltiples servicios independientes que se comunican entre sí para componer el software final. Las ventajas son varias:

- **Entendimiento**: Cada servicio es independiente, por lo que el tamaño es considerablemente menor que el que compondría un software monolítico y es más fácil de entender, centrándose en una parte específica de la lógica de negocio.
- **Escalabilidad**: Al ser un módulo independiente, si se requiere de una mayor cantidad de procesado para un servicio específico, se puede incrementar la capacidad de cómputo específicamente para este servicio.
- **Mantenibilidad y desarrollo**: Estos módulos independientes son más simples que el sistema completo. Por esto, su desarrollo y mantenimiento es más rápido y ágil, teniendo en cuenta solo lo que la lógica de negocio del servicio ha de hacer.
- **Control de errores**: Un error en el módulo afecta solo a este, independiente de los demás módulos que componen el sistema, por lo tanto, el riesgo de estos errores es menor.
- **Elección de tecnologías**: En un sistema basado en microservicios, los diferentes módulos pueden desarrollarse con tecnologías diferentes que se ciñan al propósito de este.

Por todo esto, los nuevos desarrollos de software priorizan el uso de microservicios como arquitectura.

2.1.3 Infraestructura

Si bien hace años era común poseer físicamente la máquina que actúa como servidor, donde los técnicos de sistemas configurarían esa máquina para que el software pudiera aceptar y responder ante peticiones, actualmente se suelen contratan máquinas virtuales, que son una virtualización del hardware de una máquina física. Estás están hospedadas en la nube y se accede comúnmente a ellas a través del protocolo SSH. No solo abstraen al desarrollador de la configuración básica de la máquina y el mantenimiento físico de la misma, si no que funcionan sobre una infraestructura optimizada para un mejor funcionamiento. Esto es, unas instalaciones de electricidad, redes, climatización, etc., mucho mejores de las que podríamos tener físicamente a un coste similar. Además de todo esto, ofrecen otro tipo de ventajas como la redundancia, es decir, si una máquina falla, habrá otra que seguirá ofreciendo el servicio con normalidad.

Además del uso de máquinas virtuales hospedadas en la nube, hay una tendencia hacía el uso de servicios de computación en la nube. Estos servicios ofrecen una abstracción superior a las máquinas explicadas anteriormente para múltiples funcionalidades que se quieran implementar, tales como servicios específicos de manejo de bases de datos, de inteligencia artificial, almacenamiento de datos, etc., de forma que el desarrollador se abstrae aún más del manejo y configuración de múltiples tecnologías, centrándose en el uso e integración de estos servicios.

Por último, hay muchos detalles en las configuraciones de los servicios ofrecidos por terceros que son críticos para su funcionamiento, y manejar muchos servicios puede llevar a confusión al desplegarlos. Es por esto por lo que se utilizan frameworks para la gestión y despliegue de proyectos basados en microservicios en la nube, como Serverless Framework o Terraform. Estas tecnologías recogen las descripciones y características de los servicios que se van a utilizar en un

proyecto, desde el montaje de la base de datos hasta la configuración de servicios de inteligencia artificial, para que el desarrollador pueda desplegarlos en la infraestructura del proveedor en la nube con un comando.

2.1.4 Aplicaciones similares

En el mercado existen diferentes aplicaciones que permiten llevar a cabo un control de asistencia en eventos. Estas aplicaciones pueden recoger funcionalidades más completas y necesarias para grandes eventos con distintos tipos de asistentes, u otras más simples.

Algunas de las aplicaciones más llamativas existentes en el mercado [2] son:

- **Attendium**: Permite la creación de listas de asistentes tanto para uno como para múltiples eventos. Avisa al gestor del evento en caso de asistentes duplicados. Ofrece una estadística básica de asistentes. Permite añadir usuarios que puedan gestionar el evento, con un sistema de roles. [3]
- **Brightweel:** Esta aplicación está destinada a los profesores y padres de alumnos de pequeña edad, donde es un mismo profesor es el que está siempre con un mismo grupo. En esta aplicación permite al profesor indicar qué alumno está cada día en clase y que los padres puedan indicar que el alumno ha llegado a través de un código QR o numérico. [4]
- **Guest Manager Check In:** Aplicación orientada a la organización profesional de eventos, donde los asistentes pueden tener diferentes etiquetas o atributos configurables por el gestor del evento. Se ofrecen estadísticas avanzadas del evento. Otras características de este producto es la capacidad de funcionar con funcionalidades adicionales como la impresión y venta de tiques para los eventos.

Otras aplicaciones como: EventXtra, EventTechGroup, FieldDrive; pueden realizar un registro a la llegada del asistente mediante reconocimiento facial, pero estando limitado a hacerse persona por persona y en el sitio donde se instale el dispositivo.

2.2 Metodologías de registro de asistencia

En esta sección se detallan las diferentes metodologías de paso de asistencia que se puedan realizar, de los cuales los más adecuados estarán presentes en la aplicación Sentinel.

Varias de las formas de registro que se describen a continuación pueden utilizar el GPS del dispositivo para cercar la posición del asistente y comprobar si realmente se encuentra en el evento.

2.2.1 Registro manual por el gestor

El registro manual se realiza de tal forma que, si el asistente está presente en una lista predefinida, el gestor al comprobar su llegada actualiza su registro para marcarle como confirmado. Por otro lado, si el asistente no se encuentra en la lista predefinida, el gestor puede darle de alta en el momento de la llegada rellenando los datos necesarios.

2.2.2 Registro manual por el asistente

El asistente, al llegar al evento, puede registrarse a sí mismo en el este a través de un dispositivo establecido para ello. El registro puede ser de dos formas: estando el asistente previamente registrado en una lista o añadiéndose a ella al acudir al evento. Esta forma de registro puede tener un uso del GPS del dispositivo para asegurar la cercanía del asistente al evento.

2.2.3 Registro por clave o pregunta

El asistente debe de poseer una clave o la respuesta a una pregunta (ejemplo: ¿Qué día de semana es hoy?). Esta clave o respuesta le servirá para acceder al registro del evento y registrarse como asistente. Esta forma de registro puede tener un uso del GPS del dispositivo para asegurar la cercanía del asistente al evento.

2.2.4 Registro libre por tiempo limitado

El gestor del evento hace posible el libre registro en el evento para los asistentes durante un tiempo limitado. Tras ese tiempo, los asistentes no podrán registrarse libremente.

2.2.5 Registro por código QR

Esta metodología de registro tiene varias formas de realización. Una primera es la generación de un código QR estático que se pueda imprimir o mostrar en una pantalla y el cual los asistentes escanean para registrarse en el evento. Otra forma es que el código QR se genere en el dispositivo en el momento que el gestor del evento desee. Por último, otra forma es que dinámicamente se generen códigos QR cada determinado tiempo. Esta última opción es muy útil para evitar el uso fraudulento de código, dado que en caso de enviar el código QR a otras personas, es probable que no les dé tiempo a escanearlo.

2.2.6 Registro por reconocimiento facial

Esta metodología de registro de asistencia se puede realizar de dos formas: individual y colectiva.

La forma individual se basa en el escaneo del rostro del asistente, ya sea realizado por sí mismo, por el gestor o por un dispositivo establecido para este propósito y que no requiera de interacción humana. Por otro lado, la forma colectiva se basa en la captura de una imagen o de un vídeo para su posterior procesado y registro de las personas cuyos rostros se hayan detectado en el vídeo o imagen.

Ambas metodologías requieren del previo registro de las características del rostro de los asistentes en la base de datos utilizada por la inteligencia artificial que analiza los rostros.

2.3 Lenguajes y frameworks para el Frontend

Sentinel es un proyecto de aplicación móvil, por lo que el Frontend será de este tipo. El lenguaje que utilizar viene definido por el framework que se pretenda usar para el desarrollo de la aplicación.

Tal y como se vio en el apartado de Estado del Arte – Desarrollo de aplicaciones, hay dos frameworks que son los más populares y utilizados en el mercado: Flutter [6] y React Native [7].

Los requisitos principales para la elección del framework son:

- Rapidez en el desarrollo de la aplicación.
- Facilidad de aprendizaje de la plataforma.
- Interfaz de usuario amigable, versátil y fácil de hacer.
- Rendimiento nativo.
- Calidad de la documentación.

Ambos frameworks son muy capaces de hacer lo que se requiere para las funcionalidades de Sentinel, teniendo cada uno ventajas e inconvenientes. React Native tiene una comunidad más

amplia y resulta más sencillo de aprender que Flutter, por el uso de Javascript. Sin embargo, la interfaz de usuario resulta más fácil de desarrollar en Flutter, debido al sistema de Widgets.

El desempate en la elección de la plataforma se ve afectado en favor de Flutter por dos factores: Es una tecnología nueva, innovadora y que muchos expertos califican como líder en el desarrollo móvil en un futuro cercano y por el uso de Widgets que facilitan la creación de la interfaz de usuario.

2.4 Lenguajes y frameworks para el Backend

Actualmente se suelen optar por diferentes formas de desarrollar un Backend. Las más típicas suelen ser el uso de un framework (Symfony, Spring, Express.js, etc.) para desarrollar el software y posteriormente desplegarlo en una máquina en la nube. Por otro lado, también se está utilizando cada vez más el uso de tecnologías Serverless ofrecidas por proveedores en la nube (Amazon Web Services, Microsoft Azure, Google Cloud, etc.) para montar una aplicación que actúe como Backend.

Para el desarrollo del Backend de la aplicación se va a utilizar tecnología Serverless. Es decir, en ningún momento vamos a configurar un servidor o una máquina para que actúe como este. Utilizaremos un proveedor de tecnologías en la nube para desplegar todos los servicios necesarios, permitiendo centrarnos en la creación de valor y desarrollo e implementación de las funcionalidades de la aplicación.

De entre los proveedores de servicios en la nube, contamos con conocimientos moderados de los servicios de Amazon Web Services [8], por lo tanto y siendo el líder en servicios en la nube, utilizaremos la plataforma de Amazon como proveedor.

La mayoría de las funciones para el servicio AWS Lambda las escribiremos en el sistema de tiempo de ejecución para javascript Node.js [9] por ser el más utilizado y que cuenta con mayor soporte por parte de la comunidad de desarrolladores, además de por todas las ventajas que ofrece como ser asíncrono, tener muchos módulos de terceros.

2.5 Herramientas para la detección facial

Existen multitud de herramientas para la detección facial de una o varias personas en una imagen:

- Crear una red neuronal para la detección facial.
- Utilizar y adaptar si es necesario una red neuronal de código abierto.
- Utilizar un proveedor en la nube que ofrezca la tecnología de reconocimiento facial.

Debido al desconocimiento de la creación y manejo de redes neuronales más allá de lo básico, y a la gran calidad que ofrecen los servicios de reconocimiento facial de proveedores en la nube, así como la facilidad e integración con otros servicios del mismo proveedor, se utilizará esta opción para realizar el reconocimiento facial en Sentinel. Como se va a utilizar únicamente los servicios de Amazon Web Services para la parte Backend de la aplicación, se utilizará el servicio Amazon Rekognition para las tareas de detección facial, al ofrecer una gran calidad y facilidad de integración con los demás servicios desarrollados y un coste por uso.

2.6 Herramientas software adicionales

Para el desarrollo del proyecto, se utilizarán algunas herramientas software y webs para el correcto desempeño de las tareas del proyecto y ayudar en las mismas.

2.6.1 Git

Git [10] es un software de control de versiones. Este software está pensado para llevar un registro de los cambios del código fuente de un programa conforme este va cambiando y el desarrollador va confirmando diferentes cambios. Además, no es un sistema lineal, sino que es capaz de tener incontables versiones del código fuente que parten desde una misma versión que pueden posteriormente ir mezclándose con otras versiones para componer el software final.

2.6.2 GitHub

Github [11] es un repositorio para software y control de versiones mediante el uso de Git. Esta plataforma permite guardar a la nube código en diferentes versiones y acceder a este mediante una interfaz gráfica por web o descargando el código a través de Git.

No solo existe GitHub para este tipo de servicios, otras herramientas como GitLab o BitBucket ofrecen alternativas interesantes para guardar código versionado en la nube. Sin embargo, dada nuestra experiencia utilizando GitHub y la posibilidad de utilizar repositorios privados para este proyecto, se utilizará esta plataforma en el proyecto.

2.6.3 Android Studio

Android Studio [12] es el entorno de desarrollo integrado oficial para el desarrollo de aplicaciones en la plataforma Android. Este software, desarrollado por JetBrains, se puede utilizar gratuitamente para el desarrollo de aplicaciones para Android. La plataforma cuenta con un emulador Android para probar en tiempo real el software desarrollado. Este software cuenta además con extensiones para desarrollar aplicaciones utilizando frameworks como Flutter o React Native.

2.6.4 WebStorm

Webstorm [13] es un entorno de desarrollo integrado para el desarrollo de aplicaciones usando el lenguaje Javascript. Al igual que Android Studio, está desarrollado por JetBrains, y comparte gran parte de las funcionalidades con este.

Ya que el código que sea necesario desarrollar para el Backend lo haremos en el entorno en tiempo de ejecución Node.js, que utiliza el lenguaje Javascript, esta herramienta nos ayudará con este propósito, al tener también soporte para código escrito para el Node.js.

A diferencia de Android Studio, esta herramienta es de pago, pero podemos usarla gratuitamente gracias a la licencia de estudiantes que se puede obtener usando las credenciales de la Universidad de Las Palmas de Gran Canaria.

2.6.5 Adobe XD

Adobe XD [14] es un programa de diseño de maquetas desarrollado por Adobe. Este programa permite crear interfaces de usuario sin código, creando flujos de usuario, diagramas funcionales y diseños en alta calidad.

Además, esta herramienta cuenta con multitud de plantillas realizadas por terceros para enriquecer el proceso de creación de la interfaz.

3. Análisis

En esta sección se detallan los casos de uso, es decir, funcionalidades para el cliente que debe de tener la aplicación. Asimismo, se explican qué entidades posee la aplicación.

3.1 Entidades

Sentinel es una aplicación en la que destacamos dos entidades, los eventos y los usuarios.

Un evento se representa como una entidad poseedora de un nombre de este, una descripción de la localización donde se va a desarrollar el evento, una fecha de comienzo y otra de final y, un dueño o gestor del evento.

Por otro lado, el usuario se representa con un nombre de usuario, nombre de la persona y apellidos de esta.

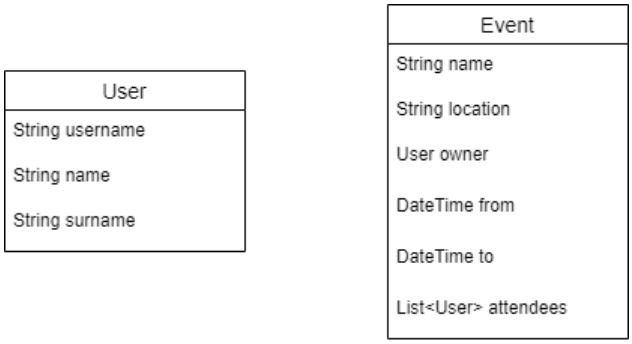


Ilustración 2: Descripción de entidades

3.2 Casos de uso

Sentinel es una aplicación para la creación de eventos para añadir usuarios y comprobar su asistencia al mismo. Dado esto, podemos diferenciar dos roles:

- **Gestor**: El gestor es el dueño de un evento, aquel que puede añadir y eliminar asistentes, además de actualizar el estado de estos. Es quien crea el evento y puede borrarlo.
- **Asistente**: Es quien es añadido al evento, ya sea por parte del gestor o el mismo a través de alguno de los métodos de registro.

Estos dos roles son claramente diferenciados, pero en la aplicación, se juntan en un mismo usuario. Esto es, un usuario puede crear un evento y añadir asistentes a este mientras que puede añadirse a otros eventos, de forma que con una misma cuenta de Sentinel se pueden realizar ambos roles.

A continuación, se muestran los diferentes casos de uso que definen las funcionalidades de las que se compone Sentinel.

3.2.1 Registro

Un usuario debe de poder registrarse en la plataforma Sentinel para crear una cuenta de usuario en la plataforma.

1. El usuario, en la pantalla inicial, pulsa en el botón habilitado para el registro en la aplicación.
2. El usuario rellena los datos requeridos para el registro.

3. El usuario pulsa el botón habilitado para completar el registro.
4. El sistema informa al usuario del éxito en la operación.
5. El sistema muestra la pantalla principal.

3.2.2 Inicio de sesión

Un usuario debe de poder iniciar sesión con su cuenta en Sentinel tras poner el usuario y contraseña adecuados en la pantalla de inicio de sesión para poder acceder a las funcionalidades de la aplicación.

1. El usuario, en la pantalla inicial, rellena los campos de usuario y contraseña y pulsa en el botón de inicio de sesión.
2. El usuario es redirigido a la pantalla principal de la aplicación, con sesión iniciada.

3.2.3 Creación de evento

El usuario debe de poder crear un evento, del cual será el gestor.

1. En la pantalla principal, el usuario pulsa en el botón habilitado para crear un nuevo evento.
2. El sistema muestra la pantalla con los campos a rellenar para la creación del evento.
3. El usuario rellena los campos para crear el evento.
4. El usuario confirma la creación del evento.
5. El sistema muestra un mensaje de éxito en la creación del evento.
6. El sistema redirige al usuario a la pantalla principal.

3.2.4 Eliminar un evento

El usuario debe de poder eliminar permanentemente un evento.

1. En la pantalla principal, el usuario realiza una pulsación larga sobre el evento que desea borrar.
2. El sistema pregunta al usuario por la confirmación de la acción.
3. El usuario confirma la acción.
4. El sistema elimina el evento.

3.2.5 Agregar un usuario a un evento

El usuario gestor de un evento debe de poder agregar un usuario a un evento del cual es gestor.

1. El usuario, en la pantalla de asistentes del evento, pulsa sobre el botón para añadir asistentes.
2. El sistema busca el usuario que desea añadir a la lista de asistentes.
3. El usuario pulsa sobre el botón para añadir ese usuario a la lista de asistentes.
4. El sistema añade el usuario a la lista de asistentes, si no lo estaba ya.

3.2.6 Eliminar un usuario de un evento

El usuario gestor de un evento debe de poder borrar un usuario de la lista de asistentes.

1. El usuario, en la pantalla de asistentes del evento, desliza hacia la izquierda sobre el registro de un asistente.
2. El sistema elimina el usuario sobre el que se ha deslizado de la lista de asistentes.

3.2.7 Envío de información del evento por e-mail

El usuario gestor de un evento debe de poder enviarse a sí mismo con el estado de la lista de asistentes.

1. El usuario, en la pantalla principal del evento, pulsa sobre el botón para enviar la información del evento por e-mail.
2. El sistema muestra un mensaje pidiendo la confirmación del usuario.
3. El usuario confirma el envío del e-mail.
4. El sistema envía al correo del usuario la información del evento.

3.2.8 Comprobación manual de asistencia

El usuario gestor de un evento debe de poder cambiar manualmente el estado de un asistente en el evento entre presente o no.

1. El usuario, en la pantalla principal del evento, pulsa sobre el botón destinado a comenzar la comprobación manual de asistentes en el evento.
2. El sistema muestra la lista de asistentes del evento.
3. El usuario marca a los diferentes usuarios como presentes o no en el evento.
4. El usuario confirma los cambios con el botón destinado a ello.
5. El sistema modifica el estado de los asistentes en el evento.

3.2.9 Comprobación de asistencia por código QR

El usuario gestor de un evento debe de poder comenzar un proceso de comprobación de asistencia mediante la generación de códigos QR dinámicamente.

1. El usuario, en la pantalla principal del evento, pulsa sobre el botón destinado a comenzar la comprobación de asistencia mediante códigos QR.
2. El sistema muestra una pantalla con un código QR que se renueva cada una cantidad determinada de segundos.

3.2.10 Comprobación de asistencia por reconocimiento facial

El usuario gestor de un evento debe de poder comenzar un proceso de comprobación de asistencia mediante la grabación de un vídeo.

1. El usuario, en la pantalla principal del evento, pulsa sobre el botón destinado a comenzar la comprobación de asistencia mediante reconocimiento facial.
2. El sistema muestra una pantalla con las instrucciones para el correcto funcionamiento del proceso de reconocimiento facial.
3. El usuario pulsa sobre el botón para iniciar la grabación del video.
4. El usuario graba un vídeo con la cara de los asistentes.
5. El sistema envía el video al sistema de reconocimiento facial.
6. Al acabar el procesado, se actualiza el evento con los nuevos usuarios cuyo rostro ha sido detectado en el vídeo.

3.2.11 Indexación del rostro del usuario

El usuario debe de poder registrar las características de su cara en el sistema para poder ser candidato a la comprobación de asistencia por reconocimiento facial.

1. El usuario accede al menú de perfil, accesible desde cualquier pantalla de la aplicación.
2. El sistema muestra la página de perfil con los detalles básicos del usuario.

3. El usuario pulsa sobre el botón habilitado para subir detalles del rostro del usuario.
4. El sistema abre la cámara del dispositivo.
5. El usuario toma una captura clara de su cara.
6. El usuario confirma la captura
7. El sistema analiza el rostro para guardar sus características en base de datos.

3.2.12 Escaneo de código QR

El usuario debe de poder escanear un código QR generado por un gestor de evento en Sentinel para añadirse al evento referido por el código.

1. El usuario accede al menú de escaneo de código QR, accesible desde cualquier pantalla de la aplicación.
2. El sistema muestra una cámara específica para la detección de códigos QR.
3. El usuario escanea un código QR generado por la aplicación.
4. El sistema añade el usuario al evento referido por el código.

4. Diseño

En esta sección del documento se detalla el proceso de diseño de software del Sentinel. Esta sección incluye:

- Diseño de la arquitectura general de la aplicación.
- Diseño de la arquitectura del Frontend.
- Diseño de la arquitectura del Backend.
- Diseño de algoritmos de funcionalidades complejas.
- Maquetado de la aplicación.

Los diferentes apartados han ido sufriendo algunos cambios conforme ha ido sucediendo el desarrollo de la aplicación, es por esto por lo que se mostrarán en su mayoría, los diseños finales para Sentinel, exceptuando en el maquetado de la aplicación.

4.1 Diseño de la arquitectura general de la aplicación

Sentinel, como la gran mayoría de aplicaciones actuales, se basa en un modelo de cliente-servidor. Esto es, la aplicación móvil desarrollada en Flutter se comunicará mediante peticiones con el servidor, el cual le responderá a las mismas. Las comunicaciones necesarias para este modelo se realizarán de forma segura mediante la utilización del protocolo HTTPS [15] (Hypertext Transfer Protocol Secure).

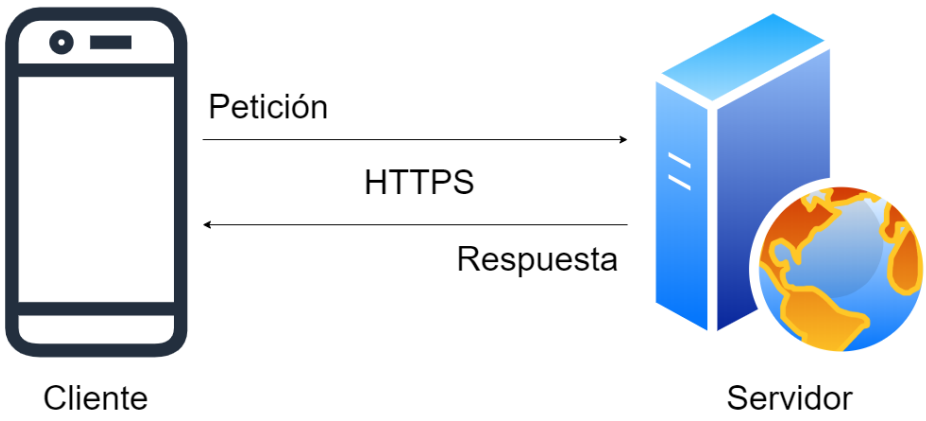


Ilustración 3: Modelo Cliente-Servidor

Las ventajas de este modelo son muchas, entre las que podemos encontrar algunas como:

- Rendimiento óptimo de la aplicación al realizar operaciones costosas en el servidor.
- Sistema centralizado con los datos en un mismo lugar.
- Eficiencia de costes al requerir menos mantenimiento.
- La capacidad de cómputo del servidor y el servidor se pueden cambiar de forma separada.
- Los servidores tienen un sistema de autentificación centralizado para que solo los usuarios autorizados tienen acceso a la información requerida.

4.2 Diseño de la arquitectura del Frontend

En esta sección se detallará la arquitectura sobre la cual se va a desarrollar la aplicación móvil. Esto es, los patrones y la estructuración del código para que sea mantenible.

Para el desarrollo de la aplicación móvil vamos a usar Flutter, un framework que se caracteriza por utilizar widgets para la creación de interfaces de usuario, que, al interactuar el usuario con estos, lanzarán eventos de diferente índole dado los cuales se desarrollará una lógica.

4.2.1 BLoC

Google ha propuesto la arquitectura oficial de Flutter llamada BLoC [16], que es la abreviatura para Business Logic Components. Esta arquitectura del software se basa en la representación de todo lo que sucede en la aplicación como un flujo continuo de eventos. Esto es, los widgets mandan eventos, manejados por el BLoC correspondiente que responderán con un estado, ante el cual la interfaz de usuario se reconstruirá con las nuevas necesidades.

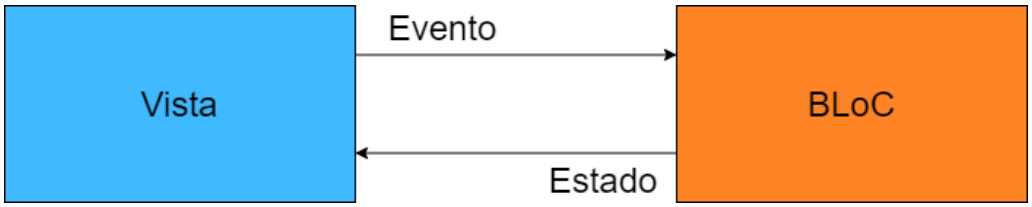


Ilustración 4: Patrón BLoC

Cada vista o pantalla de la aplicación tendrá un BLoC asociado, este no debe de manejar la lógica de varias pantallas al mismo tiempo, garantizando así el principio de responsabilidad única.

4.2.2 Repository

El patrón repository se basa en la creación de una clase intermedia entre la capa de datos de la aplicación y la capa BLoC en nuestro caso, esto es así para desacoplar la lógica del acceso a datos y la de negocio de la aplicación.

En el caso de Sentinel, los diferentes BLoC instanciarán los repositorios de las diferentes entidades para acceder así a los datos.

Al otro lado del repositorio, se encuentra la clase “proveedora” de datos, la cual realizará las diferentes llamadas al servidor con los parámetros necesarios para conseguir los datos, que serán devueltos al repositorio y posteriormente, al BLoC.

4.2.3 Singleton

Durante las diferentes etapas de la aplicación, se van a ir accediendo a los datos de una misma entidad, por ejemplo, los eventos. El envío de datos entre distintos BLoC de forma directa no debe de existir, para evitar así el acoplamiento de las diferentes clases y la instanciación de estas en vistas que no corresponden. Es por esto por lo que, para compartir datos entre BLoC se crea una clase Singleton para cada clase entidad de la aplicación que necesite que sus datos sean accesibles en diferentes partes de la aplicación, sin hacer llamadas constantes e innecesarias al servidor.

De esta forma, un BLoC podrá instanciar la clase Singleton para acceder a los datos de esta.

4.2.4 Conclusión del diseño Frontend

El desarrollo de la aplicación se basa en un patrón principal, BLoC, y esta se apoya de otros patrones de diseño, Repository y Singleton. Con todo esto, se desarrolla una aplicación móvil nativa y con una buena calidad de código, garantizando así el buen mantenimiento y evolución de la aplicación para desarrollos futuros.

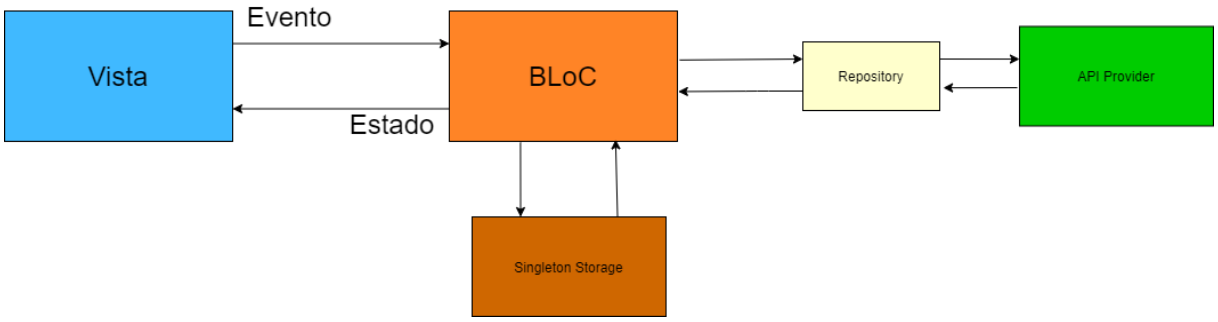


Ilustración 5: Arquitectura del Frontend

4.2.5 Maquetado de la aplicación

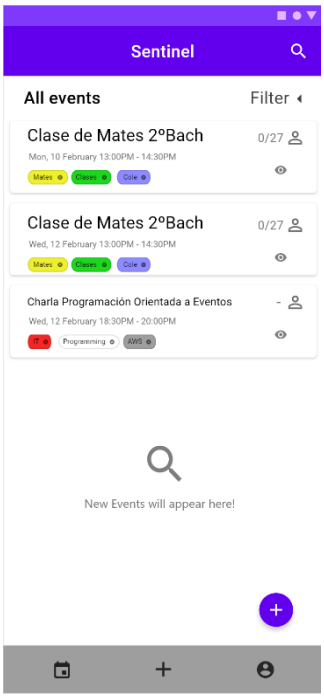
El maquetado de la aplicación fue realizado con la herramienta Adobe XD [14]. Las pantallas que se diseñaron en el maquetado han ido sufriendo variaciones conforme se ha realizado la implementación, se ha descubierto el uso de los widgets de Flutter y ha habido una limitación de tiempo. Es por esto por lo que las diferentes pantallas que se van a mostrar a continuación representan la idea inicial de la interfaz de usuario Sentinel.

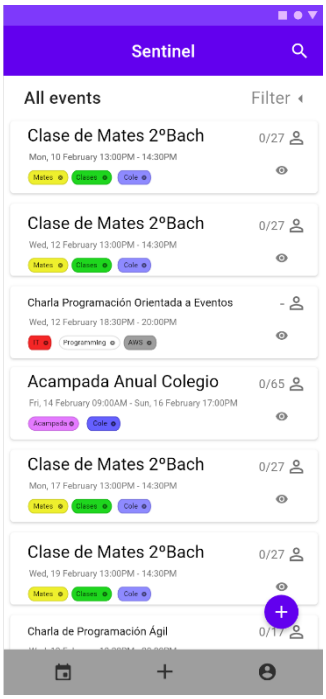
En el maquetado todas las pantallas comparten un elemento común, una barra de navegación en la parte baja de la pantalla. Esta barra sirve para que el usuario navegue por las principales pantallas de la aplicación.

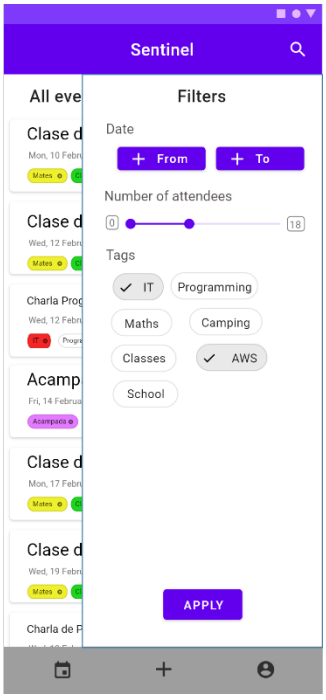
Pantalla Principal

La pantalla principal de la aplicación es la cual contendrá la lista de los eventos de los cuales el usuario es gestor. Esta pantalla contiene un botón de filtro para poder filtrar los eventos por distintos parámetros según le convenga al usuario, un botón para buscar por nombre y otro de tipo flotante para añadir un nuevo evento.

Cada tarjeta que contiene la información de un evento tiene una acción de pulsación corta asociada para navegar a los detalles del evento.







Detalles del evento

La pantalla de detalles del evento contiene la información general relevante del evento; título, fecha de inicio y de final, descripción.

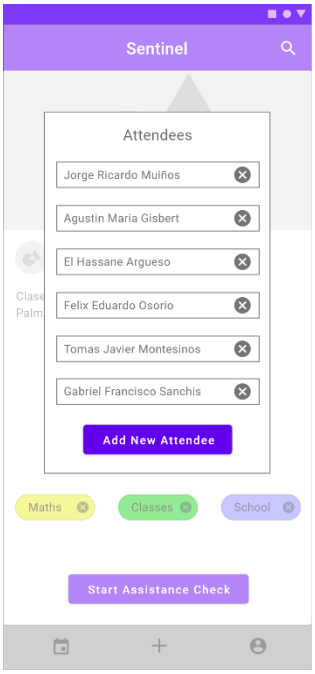
Además, esta pantalla ofrece acceso a la lista de asistentes del evento y a los diferentes métodos de comprobación de asistencia.

Por otro lado, se muestran las etiquetas que caracterizan al evento, así como la posibilidad de añadir nuevas y eliminar algunas de las existentes.

Por último, en esta pantalla también se muestra la posibilidad de generar un archivo PDF (descartada esta opción en la implementación final) y de enviar el un PDF por e-mail (sustituida esta opción por un e-mail sin ficheros adjuntos).

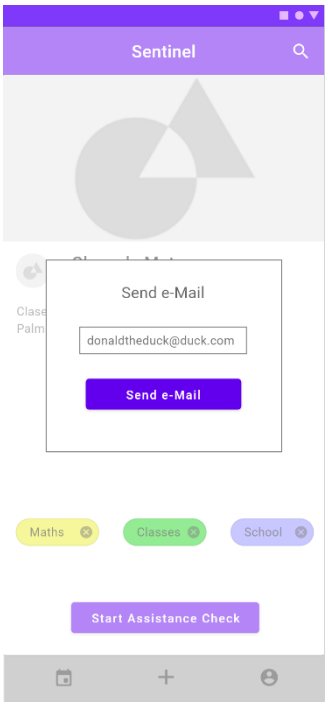
Lista de asistentes

La pantalla de la lista de asistentes se define por sí sola, es una vista de los asistentes que se encuentran registrados en el evento, ya sea como presentes o no en el mismo, en el momento en el que se comprueba la lista. En esta vista también existe un botón para añadir nuevos asistentes al evento.



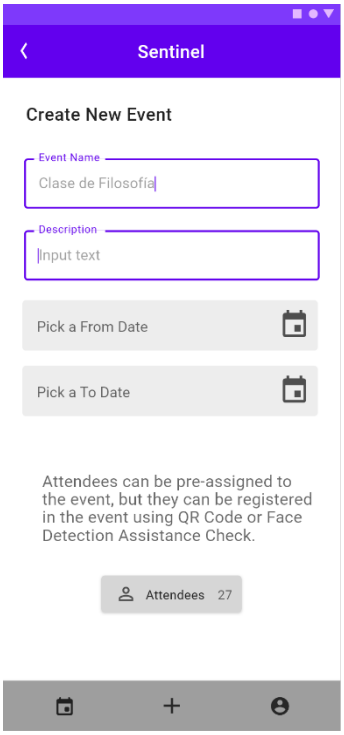
Enviar evento por e-mail

Esta pantalla consiste en un campo de texto donde el usuario introduce la dirección de correo electrónico a la cual se quiere enviar el resumen de la asistencia del evento, una vez introducido se pulsa en el botón de enviar e-mail y este mandado al correo escrito.



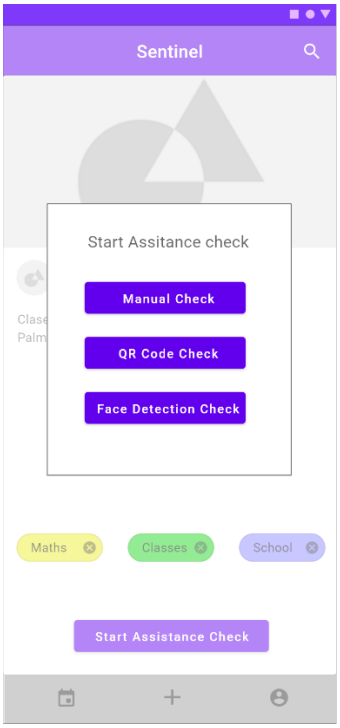
Creación de evento

Esta vista muestra los campos necesarios para la creación de un evento en Sentinel. Estos consisten en dos campos de texto normales y dos selectores de fechas. Adicionalmente se muestra una opción para añadir asistentes directamente en la creación del evento.



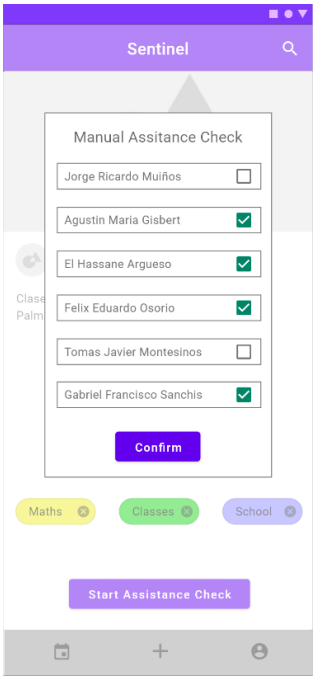
Comenzar comprobación de asistencia

Esta vista muestra las diferentes opciones que se dan para la comprobación de asistencia.



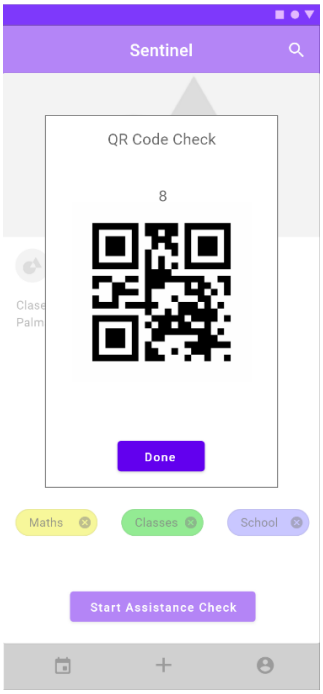
Comprobación manual de asistencia

Esta vista muestra la lista de asistentes del evento, acompañado cada registro con un elemento "checkbox" que permitirá al gestor del evento marcar a cada asistente como presente o no en el mismo. Una vez finalizados los cambios, se pulsa en el botón de confirmación y serán guardados.



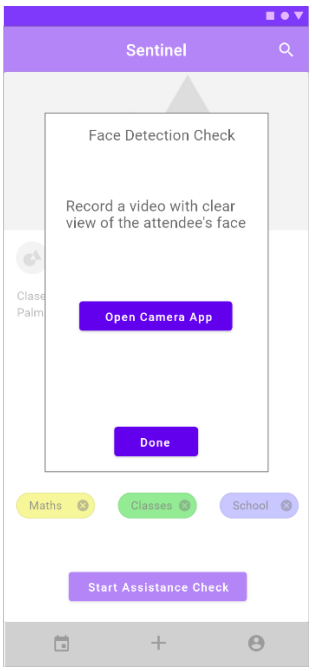
Comprobación de asistencia mediante códigos QR

Esta vista tiene como elemento principal el código QR generado para ser escaneado por otros usuarios. Asimismo, se muestra el contador de los segundos que quedan para que el código sea inválido y se reemplace por otro. Por último, se muestra un botón para finalizar la comprobación.



Comprobación mediante reconocimiento facial

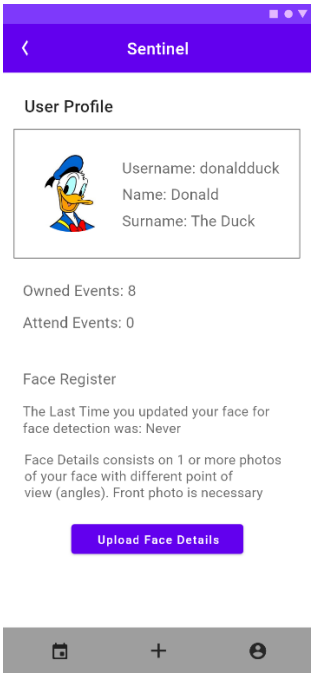
Esta vista muestra, principalmente, el botón mediante el cual se abrirá la cámara para comenzar con la grabación de los rostros de los asistentes.



Perfil

Esta vista muestra los datos principales del usuario; nombre, apellidos, nombre de usuario. Asimismo, se muestra la cantidad de eventos de los cuales es gestor, y el número de cuantos es participante.

Por otro lado, se muestra cuando fue la última vez que se subió una imagen del rostro del usuario y se añade la posibilidad de que el usuario suba los detalles de su rostro, lo cual consiste en una imagen que será enviada al Backend para que la analice.



4.3 Diseño de la arquitectura del Backend

En esta sección se va a detallar la arquitectura con la cual se configura el Backend, esta arquitectura está basada completamente en microservicios y en tecnología Serverless [17]. Es decir, se van a utilizar servicios del proveedor en la nube Amazon Web Services [8] para configurar todo el lado servidor de Sentinel. Con esto conseguiremos unas ventajas clave:

- **Coste por demanda**: Solo se paga por el uso de los servicios conforme se usen. Si un servicio no se usa, no se paga por este, aunque esté configurado.
- **Escalabilidad**: Los microservicios pueden escalar de forma individual conforme su uso aumente, y viceversa. Esto se hace además de forma automática por parte del proveedor.
- **Modularidad**: Cada servicio funciona y se configura por separado. Estos están conectados entre sí por medio de eventos que los activan.

A continuación, se detallan los servicios de Amazon Web Services que se van a utilizar en para el desarrollo del Backend.

4.3.1 Amazon API Gateway

Este servicio ofrece la creación de una puerta de entrada para desarrollar una API y gestiona todas las tareas implicadas en la aceptación y procesamiento de las llamadas simultáneas a la API. Algunas de las tareas que gestiona este servicio son: Administración del tráfico, compatibilidad con CORS (Cross-origin resource sharing), control de acceso y autorizaciones.

Este servicio será utilizado por Sentinel para vincular llamadas a las API con funciones administradas por el servicio Lambda, que llevará a cabo la lógica necesaria requerida por el cliente.

4.3.2 Amazon Lambda

El servicio AWS Lambda permite la creación de funciones en la nube sin utilizar servidores. El desarrollador crea una función lambda, carga el código en esta y el servicio se encarga de ejecutar y escalar la función según eventos estén configurados para usarla y haya más o menos ejecuciones de esta.

El servicio tiene algunos valores básicos de configuración, como elegir la cantidad máxima de memoria RAM que se le asigna a la función o el tiempo máximo de ejecución.

AWS Lambda es el centro de nuestro Backend, es donde reside la lógica en las diferentes llamadas a varios servicios. Este servicio se encargará de escribir los registros necesarios, acceder a base de datos, etc.

4.3.3 Amazon Dynamo DB

Dynamo DB es un servicio de bases de datos NoSQL que ofrece un alto rendimiento y una gran escalabilidad para trabajar con cualquier volumen de peticiones. Este servicio permite al desarrollador abstraerse de la configuración y el despliegue de la base de datos. Además, cumple con otros requisitos necesarios como garantizar la seguridad de los datos, la redundancia de estos, disponibilidad, etcétera.

Las tablas de Dynamo DB se crean con una clave de partición (identificador) o una clave compuesta por la de partición y de ordenación.

4.3.4 Amazon Simple Storage Service (S3)

El servicio Amazon Simple Storage Service, también conocido como S3, es un servicio de almacenamiento de objetos que ofrece una alta escalabilidad, disponibilidad de los datos, redundancia de estos y una alta seguridad. Esto es, se pueden guardar objetos como imágenes, vídeos o código fuente de cualquier tamaño, pagando solo por el espacio consumido y las peticiones de acceso al recurso. Por otro lado, el servicio ofrece un abanico de eventos para activar otros servicios cuando una acción sucede en S3.

El servicio funciona mediante la creación de "Buckets" o contenedores donde se ubicarán clasificados los diferentes archivos según el criterio del desarrollador. Por ejemplo, todos los ficheros de código fuente de Sentinel podrían ir en un Bucket llamado "Sentinel-source-code".

4.3.5 Amazon Cognito

Amazon Cognito es un servicio que ofrece autentificación para cualquier tipo de aplicación. Este servicio se encarga de manejar los diferentes grupos de usuarios que queramos para la aplicación, gestionando registros, inicios de sesión, control de acceso a recursos según roles o uso de identificación por medio de terceros como Google o Facebook, entre otros.

4.3.6 Amazon IAM

Este servicio, cuyas siglas significan "Identify and Access Management", permiten la creación de usuarios y roles para que estos tengan accesos limitados a los recursos ofrecidos por AWS.

Amazon recomienda desarrollar servicios utilizando el principio de menor privilegio (Least Privilege Principle), el cual se basa en que un usuario o rol tenga el mínimo permiso posible para que pueda realizar sus funciones de forma que no pueda ver o manejar, errónea o intencionadamente, los diferentes recursos de la plataforma.

4.3.7 Amazon Simple Email Service (SES)

Amazon SES es un servicio que permite la gestión de correos electrónicos que permite a los usuarios el envío de correos en su aplicación. Este servicio se puede integrar, entre otras formas, programáticamente en una API ofreciendo las credenciales adecuadas o realizando llamadas con los roles adecuados desde funciones Lambda.

4.3.8 Amazon CloudWatch

Este servicio ofrece el monitoreo y registro de las actividades de los diferentes servicios de Amazon. En él, se pueden guardar registros de, por ejemplo, cada ejecución de una función lambda.

4.3.9 Amazon Rekognition

Amazon Rekognition es un servicio que ofrece el análisis de vídeos e imágenes para su procesado con inteligencia artificial. Este servicio ofrece al desarrollador el uso de inteligencia artificial altamente precisa, rápida y escalable sin tener que desarrollar nada, incluso sin conocer acerca de inteligencia artificial.

4.3.10 Amazon Elastic Beanstalk

Amazon Elastic Beanstalk es un servicio que permite a un desarrollador implementar y escalar servicios web desarrollados en diferentes tecnologías con el objetivo de que el servicio se encargue de absolutamente todo lo referente a la configuración del servidor y despliegue del servicio web. Del mismo modo, aunque sea Amazon quien automáticamente administre todo el

sistema, el desarrollador podrá acceder a los recursos que use el servicio web en cualquier momento.

Este servicio utiliza el servicio EC2 para funcionar, y es este el que establece los costes. No se cargan tarifas adicionales por Elastic Beanstalk; solo paga por los recursos de AWS que necesite para almacenar y ejecutar las aplicaciones.

4.3.11 Amazon Elastic Compute Cloud (EC2)

Elastic Compute Cloud, conocido como EC2, es un servicio que proporciona acceso a máquinas virtuales en la nube. Estas máquinas se montan con un hardware virtualizado, modificable en cualquier momento.

Amazon ofrece diferentes tipos de instancias o máquinas virtuales que satisfacen casi por completo las necesidades de los diferentes desarrolladores.

4.3.12 Infraestructura completa

Los servicios descritos anteriormente ofrecen todos unos costes bajo demanda, una alta escalabilidad y al ser independientes y modularidad adecuada para cualquier aplicación.

Todos estos servicios deben de conectarse entre sí mediante eventos o invocaciones directas de los mismos. A continuación, se muestra un diagrama completo de la infraestructura de servicios que compondrán el Backend de Sentinel.

El servicio Elastic Beanstalk surge por la necesidad de desplegar una API para el procesado de los vídeos para extraer imágenes posteriormente mediante el uso de Amazon Rekognition. Esto es así porque las funciones Lambda establecen un límite de ficheros, incluyendo dependencias, que se pueden configurar en la misma, siendo excedido para la función que analiza extrae imágenes de los videos. Por ello, hubo que crear una API que se encargase de procesar dichos vídeos. Esta API está desarrollada con el framework Flask y se usó el servicio mencionado anteriormente para desplegarla de una manera rápida y sencilla.

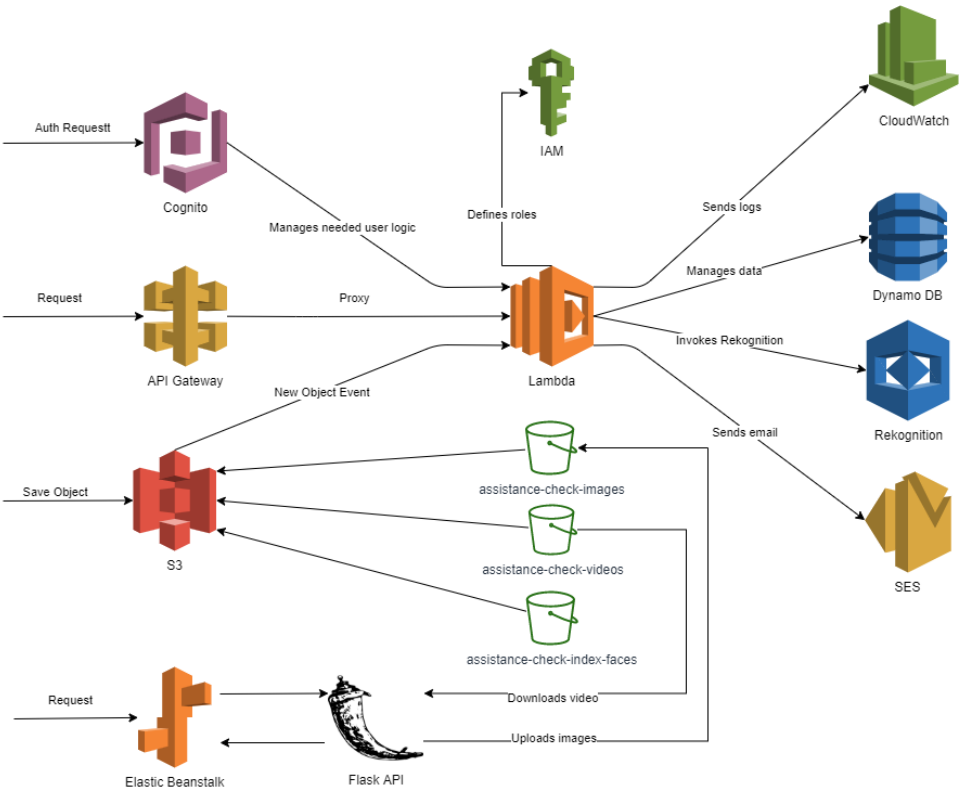


Ilustración 6: Arquitectura de la infraestructura

Como se puede apreciar, el cliente puede acceder al uso de los servicios a través de carga de objetos en S3, realizando peticiones a API Gateway o usando la autentificación que ofrece Sentinel al cliente. Todas estas acciones desembocan en eventos que activan funciones lambda que realizarán la lógica de negocio adecuada accediendo a los diferentes servicios, como son SES, Rekognition o DynamoDB. Todas las funciones lambda realizarán un registro de su estado en el servicio CloudWatch.

Como se puede apreciar, el servicio S3 se encuentra configurado con tres contenedores diferentes, cada uno con un propósito.

- assistance-check-videos: El cliente al realizar un vídeo para realizar la comprobación de asistencia, guarda el vídeo en este contenedor.
- assistance-check-images: La lambda encargada del procesado del vídeo, extrae todas las imágenes de caras de este y las introduce en este contenedor.
- assistance-check-index-faces: El usuario que desee registrar las características de su rostro en el sistema, cargará la imagen en este contenedor, donde un evento activará la función Lambda que procesará la imagen.

4.4 Algoritmos de funcionalidades complejas

Algunas de las funcionalidades clave de Sentinel requieren de un estudio previo de los diferentes flujos que se realizarán por el sistema para que su posterior implementación resulte satisfactoria.

4.4.1 Comprobación de asistencia mediante código QR

En este apartado se muestran los diagramas de flujo realizados para el diseño de la funcionalidad de comprobación de asistencia utilizando códigos QR.

Generación de códigos QR por parte del gestor en el Frontend

El gestor de un evento desea comenzar la comprobación de asistencia.

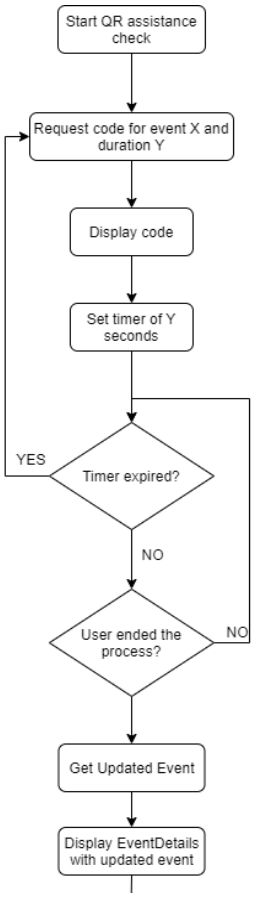


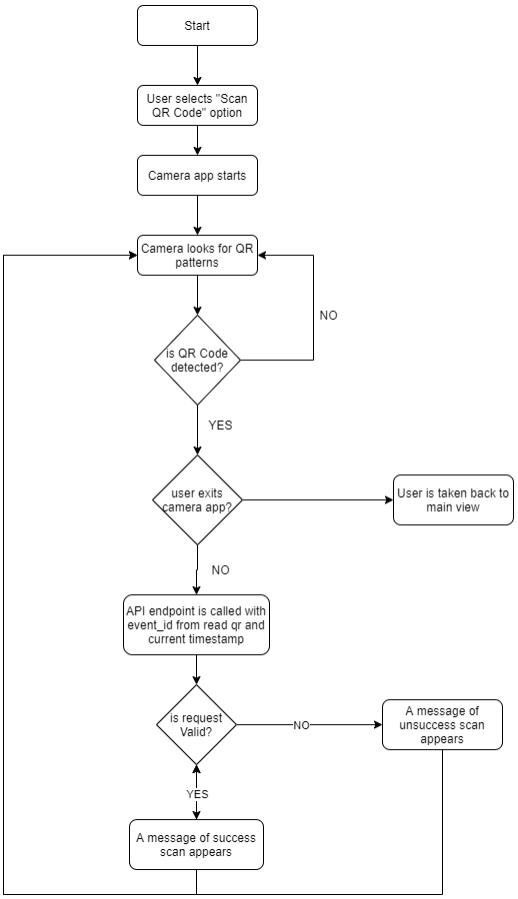
Ilustración 7: Generación de código QR desde la aplicación

Este diagrama muestra el flujo que debería de realizarse en la aplicación para iniciar una comprobación de asistencia mediante códigos QR.

1. En primer lugar, el usuario debe de pulsar el botón en la pantalla principal del evento para comenzar la comprobación de asistencia mediante este método.
2. La aplicación realizará una petición al Backend de un código para el evento indicado y con una duración establecida por la aplicación.
3. Una vez recibida la respuesta, la aplicación generará un código QR con la información recibida y este será mostrado en la pantalla del dispositivo.
4. A la vez que este último paso es realizado, la aplicación iniciará un temporizador con una duración igual a los segundos de duración de la petición anterior.
5. Cuando este contador finaliza, el código QR ya no es válido. La aplicación vuelve a realizar las acciones a partir del paso 2.
6. Cuando el usuario termina el proceso de comprobación de asistencia, se vuelve a la página principal del evento, donde se pedirá al servidor el evento actualizado.
7. Se muestra el evento actualizado.

Escaneo de códigos QR por parte del usuario asistente al evento

El usuario asistente al evento escanea un código QR para registrar su asistencia al evento.



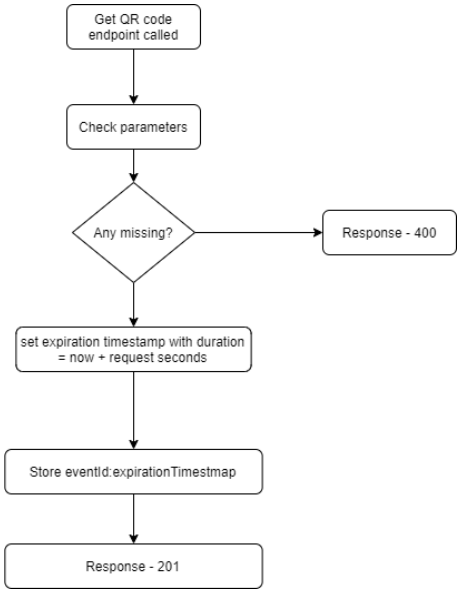
Este diagrama de flujo muestra el proceso desde que el usuario selecciona la opción de escanear un código QR hasta que el sistema informa de un éxito o error en la operación. O hasta que el usuario sale de la opción.

1. El usuario selecciona la opción de escanear código QR desde la aplicación.
2. El sistema inicia la cámara que comenzará a buscar por un patrón QR.
3. Si se detecta un código, este se leerá y se realizará una petición al Backend con la marca de tiempo en la que se realizó la detección del código y el evento asociado al código.
4. La aplicación esperará a una confirmación o denegación de la petición.
5. Si la petición es válida, un mensaje de éxito informa al usuario de su registro en el evento. Si la petición no es válida, está fuera de tiempo, un mensaje de error informa al usuario.
6. El sistema sigue buscando códigos QR hasta que el usuario finaliza la operación y es llevado a la vista principal de la aplicación.

Ilustración 8: Escaneo de código QR desde la aplicación

Generación de códigos QR en el Backend

El Backend de Sentinel recibe una petición de generación de código para la comprobación QR.

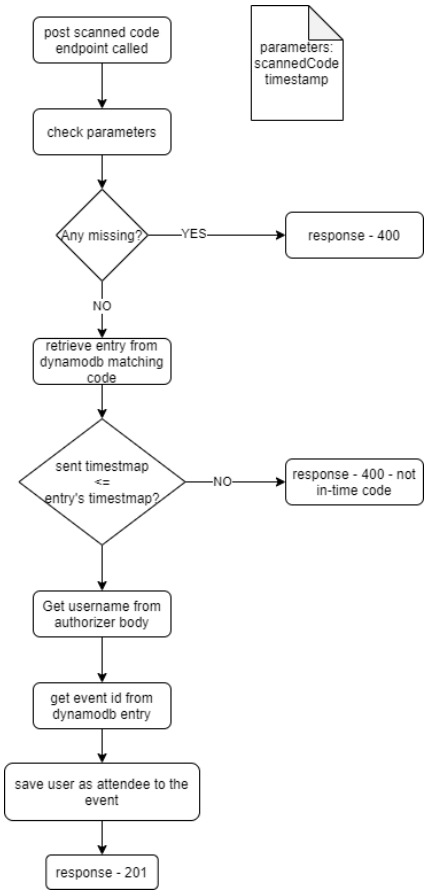


1. El Backend recibe una petición para la generación de un código que sirva para la comprobación QR.
2. El sistema comprueba que los parámetros recibidos en la petición no están incompletos.
3. Se establece una marca de tiempo para la expiración del código.
4. Se guarda el código con el identificador del evento y la marca de tiempo de expiración de este y un identificador único para el código.
5. Se devuelve el código generado al cliente.

Ilustración 9: Generación de código

Comprobación en el Backend del código escaneado

El Backend de Sentinel recibe una petición de registro con un código escaneado.



1. El Backend recibe una petición para la comprobación de un código previamente escaneado por la aplicación para el registro de asistencia.
2. El sistema comprueba que los parámetros están completos.
3. El sistema recupera el registro de base de datos cuyo identificador es el código recibido.
4. Si la marca de tiempo de la petición es menor o igual que la del registro recuperado de base de datos, la petición es válida.
5. Se recupera el usuario que realizó la petición.
6. Se recupera el evento asociado al código.
7. Se guarda en base de datos el usuario como asistente del evento.
8. Se devuelve una respuesta de éxito al cliente.

Si la petición está malformada o la marca de tiempo enviada por el cliente está fuera de tiempo, se informará a este con un mensaje de error.

Ilustración 10: Comprobación de código en Backend

4.4.2 Comprobación de asistencia mediante reconocimiento facial

Este método de comprobación de asistencia ha sufrido varios cambios, los cuales se mostrarán en la sección de implementación. En esta sección se muestran los diagramas finales. El primer paso para esta funcionalidad es grabar un vídeo con la aplicación que posteriormente es cargado en un Bucket de Amazon S3, donde se dispara un evento. Este primer paso es omitido en los diagramas por su sencillez.

De esta forma, se mostrarán dos diagramas diferenciados. Un primer diagrama mostrará cómo funciona la extracción de imágenes de los vídeos usando la API desarrollada con Flask y que funciona gracias al servicio Elastic Beanstalk, y la otra parte mostrará como esas imágenes son enviadas a Amazon Rekognition para su análisis.

Extracción de imágenes

La extracción de imágenes se basa en el análisis de los diferentes fotogramas del vídeo para detectar en rostros en estos. Una vez detectado el rostro, se analizan las características y se contrastan con las ya analizadas en este proceso, para evitar duplicidad de caras.

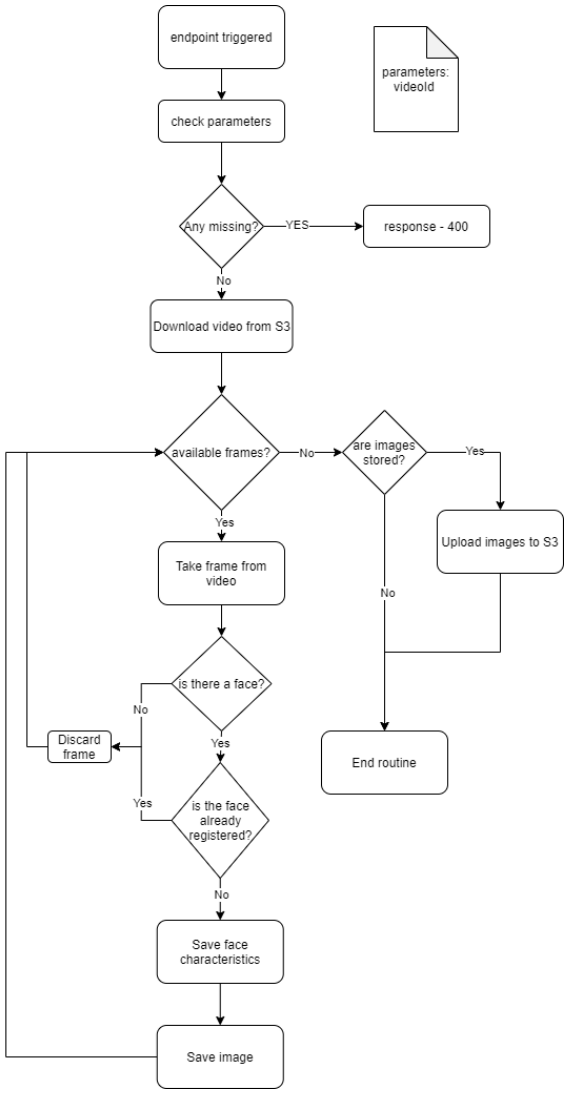
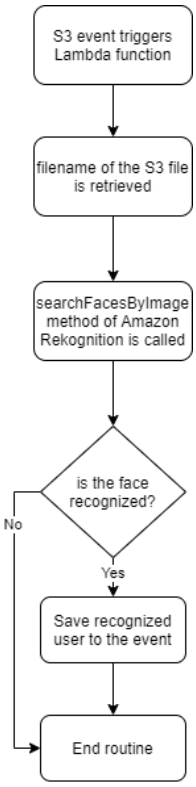


Ilustración 11: Análisis del vídeo

1. La API recibe una petición para analizar un video.
2. El sistema comprueba que los parámetros están completos.
3. El sistema descarga el vídeo a analizar desde S3.
4. Se inicia un bucle que itera sobre los fotogramas del vídeo.
5. Se recupera un fotograma del vídeo
6. Si hay un rostro en este, se comprueba si ya ha sido analizado.
7. Si no ha sido analizado, se guardan las características del rostro para comprobaciones futuras y se guarda una imagen del rostro.
8. Se vuelve al punto 5.
9. Cuando se acaban los fotogramas a analizar del vídeo, si hay imágenes guardadas en memoria, se suben estas a S3.

En caso de que en un fotograma no se encuentre un rostro o el mismo ya se haya registrado, se descarta el fotograma y se vuelve al punto 5.

Análisis de imágenes



Cuando finaliza el procedimiento de extracción de imágenes, estas son cargadas en un contenedor de Amazon S3 creado para este propósito. Cuando un archivo es cargado, el contenedor dispara un evento para que se active una función lambda que actúe en consecuencia.

1. La función se activa mediante un evento de S3 tras haber puesto una imagen en el contenedor.
2. La función recupera de la información del evento que la ha activado, el nombre del fichero que provocó el evento.
3. La función prepara los parámetros y llama a la función de Amazon Rekognition que analiza rostros en busca de alguno registrado en la colección de caras.
4. Si se detecta un usuario registrado en la imagen enviada, se asigna el usuario como asistente que ha acudido al evento.

Si no se detecta ningún usuario registrado en la imagen, la función termina su funcionamiento.

5. Implementación

En esta sección del documento se describe la fase de implementación del proyecto. En esta, se explicarán las dificultades por las que se ha pasado durante el proceso, así como detalles relevantes en la implementación de este.

Los casos de uso realizados en la sección de Análisis y los diferentes diseños realizados en la sección posterior, sirven para agilizar este proceso, minimizando las posibles dudas acerca de las funcionalidades y su diseño.

De esta forma, en esta sección vamos a ver:

- Detalles básicos de la implementación del Frontend; BLoC, Singleton, Repository, librerías, posibles dificultades encontradas.
- Detalles de la implementación de los servicios del Backend, así como las dificultades encontradas.
- Proceso de pruebas y validaciones

5.1 Implementación del Frontend

La implementación de la parte Frontend del proyecto Sentinel se basa en la creación de una aplicación nativa mediante el framework Flutter. Este framework permitirá desarrollar la aplicación que posteriormente se traducirá a lenguaje nativo de Android y iOS. En nuestra implementación, se va a generar la aplicación únicamente para Android dado que, aunque el mismo código sirva para ambos sistemas, la generación de la aplicación en iOS requiere de un ordenador con el sistema operativo Mac y la herramienta Xcode.

Al igual que muchos otros frameworks, Flutter permite la adición de dependencias de terceros que aportan funcionalidades a la aplicación. Estos módulos se irán explicando según corresponda en cada apartado.

5.1.1 Widgets

Los Widgets son componentes gráficos configurables en los que se basa Flutter para el desarrollo completo de la interfaz de usuario. Estos componentes son altamente versátiles y se pueden anidar con otros para formar estructuras gráficas complejas.

Hay dos tipos de estado, Stateless Widget y Stateful Widget, sin estado y con este respectivamente. Cuando un widget con estado cambia, se puede informar a este del cambio de estado para que se recargue, mostrando los nuevos datos si los hay.

Para la implementación de la interfaz de usuario de Sentinel, se han utilizado una gran variedad de Widgets, anidándolos en muchos casos para formar diferentes componentes.

5.1.2 BLoC

La parte más importante del Frontend de la aplicación es la correcta implementación del patrón BLoC [16], el cual definirá la forma de comportarse y compartir información que tendrá la aplicación.

Para hacer más fácil la implementación, hacemos uso del paquete “flutter_bloc”.

Para explicar la implementación del patrón BLoC, se va a tomar como ejemplo uno de los muchos que han sido desarrollados. El bloque escogido es el correspondiente a la creación de un evento.

Un BLoC recibe un evento lanzado desde la capa de interfaz de usuario o vista, este lo captura y actúa en consecuencia para responder con un estado. Estos pueden ser cualquier tipo de variable; cadenas de texto, enteros, instancias de clases, etcétera.

En nuestro caso y para lograr una mayor consistencia en la creación, envío y recepción de eventos y estados, hemos decidido implementarlos como clases que extienden de una clase padre genérica para eventos y otra para estados.

En la siguiente imagen podemos apreciar la clase "CreateEventState", abstracta y con cuatro clases herederas de esta:

- **InitState**: Clase que define un estado inicial, cuando el BLoC aún no ha recibido ningún evento.
- **LoadingState**: Estado que se suele lanzar cuando el BLoC está procesando un evento. Normalmente se lanza nada más capturar un evento, para que se muestre al usuario una vista indicando que se está procesando una petición.
- **EventCreationSuccessful**: En el caso del BLoC de ejemplo, crear un evento, este estado indicaría que se ha creado correctamente el evento.
- **EventCreationUnsuccessful**: Indica que un evento no se ha podido crear.

```
abstract class CreateEventState {}

class InitState extends CreateEventState {}
class LoadingState extends CreateEventState {}
class EventCreationSuccessful extends CreateEventState {}
class EventCreationUnsuccessful extends CreateEventState {}
```

Ilustración 12: CreateEventState

Si la imagen anterior mostraba los posibles estados para el BLoC, la siguiente imagen muestra los eventos que este puede recibir y actuar en consecuencia.

La clase abstracta CreateEventEvent tiene una sola clase hija, "EventCreationsubmittedEvent", que como su mismo nombre indica, es un evento mandado desde la interfaz de usuario indicando que se quiere crear un evento y, por tanto, esta clase contendrá los datos necesarios para esta operación, no pudiendo ser nulos.

```
abstract class CreateEventEvent {}

class EventCreationSubmittedEvent extends CreateEventEvent{
  String name;
  String location;
  DateTime startDate;
  DateTime endDate;
  EventCreationSubmittedEvent({@required this.name, @required this.location,
    @required this.startDate, @required this.endDate}) :
        assert(name != null, location != null),assert(startDate != null, endDate != null);
}
```

Ilustración 13: CreateEventEvent

Una vez descritas las clases que hacen posible la recepción y envío de eventos y estados, se muestra el BLoC que las utilizará.

Como podemos ver en la imagen, CreateEventBloc extiende de la clase Bloc, que viene definida en el paquete “flutter_bloc”. Los métodos más importantes son:

- **initialState**: Este método “getter” sirve para obtener la definición del estado inicial cuando el BLoC es creado, pero no se le ha lanzado ningún evento.
- **mapEventToState**: Este método es el que se ejecuta cuando un evento es añadido al BLoC. Recibe un evento de la clase que hayamos definido para este propósito y tiene como característica que devuelve un “stream” o en español, un torrente o flujo de datos de la clase que hayamos definido para los estados. De esta forma, nada más entrar en el método, podemos devolver un estado para que la vista informe al usuario que está habiendo un proceso. Posteriormente, cuando corresponda, se pueden devolver otros estados, como que un evento ha sido creado correctamente.

Podemos apreciar también como se hace uso de, en el caso de la imagen, el repositorio de eventos para hacer llamadas a la API.

```
class CreateEventBloc extends Bloc<CreateEventEvent, CreateEventState> {
  final EventRepository eventRepository = EventRepository();

  @override
  void onTransition(
      Transition<CreateEventEvent, CreateEventState> transition) {
    super.onTransition(transition);
  }

  @override
  CreateEventState get initialState => InitState();

  @override
  Stream<CreateEventState> mapEventToState(CreateEventEvent event) async* {
    yield LoadingState();
    if(event is EventCreationSubmittedEvent){
      int statusCode = await eventRepository.createEvent(event.name, event.location, event.startDate, event.endDate);
      if(statusCode == 200 || statusCode == 201) yield EventCreationSuccessful();
      else yield EventCreationUnsuccessful();
    }
  }
}
```

Ilustración 14: CreateEventBloc

La última pieza para completar la implementación de un BLoC es lograr que las vistas se refresquen cuando llega un nuevo estado para poder así actuar en consecuencia.

Para ello, el paquete “flutter_bloc” nos otorga un método, “BlocBuilder”. Este método recibe dos parámetros obligatorios, una instancia de un bloc y una función que escucha los estados recibidos por este.

Como podemos ver en la siguiente imagen, la función asociada al widget recibe un estado y cada vez que este cambia se ejecuta para actuar en consecuencia mostrando las vistas adecuadas para cada caso.

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: SentinelAppBar().buildAppBar(),
    body: BlocBuilder(
        bloc: _createEventBloc,
        builder: (context, state) {
          if (state is LoadingState)
            return Status().buildLoadingView();
          else if (state is EventCreationSuccessful) {
            SchedulerBinding.instance
                .addPostFrameCallback((_) => showFloattingMessage("Event Created Successfully"));
            return buildCreateEventBody();
          } else if (state is EventCreationUnsuccessful) {
            SchedulerBinding.instance.addPostFrameCallback((_) =>
                showFloattingMessage("Oops!! There was a problem while creating the event", isSuccess: false));
            return buildCreateEventBody();
          }
          return buildCreateEventBody();
        }), // BlocBuilder
  ); // Scaffold
}
```

Ilustración 15: BlocBuilder

5.1.3 Capa de usuario

Para conseguir la correcta implementación de la capa de usuario en el Frontend, es decir, la correcta conectividad de la aplicación con el Backend, hacemos uso de un módulo que ofrece integración de Flutter con las funciones de Amazon Cognito, llamado "amazon_cognito_identity_dart_2".

Este módulo no es oficial por parte de Amazon, pero se basa en el módulo "amazon-cognito-identity-js" de Node.js para ser adaptado a su uso en Dart.

Una vez añadida la referencia al módulo, ofrece las funcionalidades de inicio de sesión, registro, cambio de contraseña, autentificación en dos pasos, etcétera., que aporta el servicio de Amazon. Para usar estas funcionalidades, tan solo hay que importar las funciones del módulo en el repositorio donde se quiera usar.

5.1.4 Carga de archivos en Amazon S3

La carga de archivos en Amazon S3 es una tarea que se puede implementar utilizando las llamadas API en las que se basa la conectividad de la aplicación. Sin embargo, algunos de estos archivos pueden ser pesados, por lo que resulta conveniente utilizar otra vía de carga de estos. Dart tampoco cuenta con soporte nativo para Amazon S3. Sin embargo, el módulo usado para la integración de la capa de usuario en el Frontend también ofrece funcionalidades para la carga de ficheros en Amazon S3.

Para la implementación de la funcionalidad, hay que poseer las claves tanto de acceso como secreta que permiten el uso del servicio de Amazon, esto es así porque no es una petición que se realice con el autorizador propio de Cognito, pese a que se utiliza el mismo módulo. Junto con estas claves y el enlace que apunta al contenedor de los archivos que queramos cargar, se implementa la funcionalidad.

```dart
Future<int> uploadFile(File file, String username) async {
  const _accessKeyId = 'XXXXXXXXXXXXX';
  const _secretKeyId = 'XXXXXXXXXXXXxX';
  const _region = 'eu-central-1';
  const _s3Endpoint = 'https://assistance-check-index-faces.s3.eu-central-1.amazonaws.com';

  final stream = http.ByteStream(DelegatingStream.typed(file.openRead()));
  final length = await file.length();

  final uri = Uri.parse(_s3Endpoint);
  final req = http.MultipartRequest("POST", uri);
  final multipartFile = http.MultipartFile('file', stream, length, filename: path.basename(file.path));
  final policy = Policy.fromS3PresignedPost(username, 'assistance-check-index-faces', _accessKeyId, 15, length, region: _region);
  final key = SigV4.calculateSigningKey(_secretKeyId, policy.datetime, _region, 's3');
  final signature = SigV4.calculateSignature(key, policy.encode());

  req.files.add(multipartFile);
  req.fields['key'] = policy.key;
  req.fields['acl'] = 'public-read';
  req.fields['X-Amz-Credential'] = policy.credential;
  req.fields['X-Amz-Algorithm'] = 'AWS4-HMAC-SHA256';
  req.fields['X-Amz-Date'] = policy.datetime;
  req.fields['Policy'] = policy.encode();
  req.fields['X-Amz-Signature'] = signature;

  try {
    final res = await req.send();
    await for (var value in res.stream) {
      print(value);
    }
    return res.statusCode;
  } catch (e) {
    print(e.toString());
  }
  return 0;
}
```

Ilustración 16: Carga de ficheros en S3

5.1.5 Reconocimiento facial

El reconocimiento facial es una tarea que en la parte Frontend resulta trivial, pero por ser una funcionalidad clave de Sentinel, se describe aquí su implementación.

La tarea de la aplicación en este proceso es la de permitir al usuario acceder al menú donde activar la opción para proceder a la comprobación de asistencia por este método y posteriormente ponerlo en marcha.

La primera parte de este proceso es una navegación por las diferentes vistas y opciones de la aplicación. La segunda utiliza un módulo, “image_picker”, utilizado para dar acceso a la aplicación a la cámara, pero, a diferencia de la integración nativa de la cámara en Flutter, esta solo da la opción al usuario de grabar vídeo.

Previo a la grabación de vídeo se muestran unas instrucciones de cómo realizarlo de forma correcta y, una vez terminada la grabación de vídeo la aplicación llevará al usuario a esta misma vista previa al paso de asistencia. Esta decisión hace que para el usuario una vez confirmada la grabación de vídeo, la aplicación no está realizando ninguna acción. Sin embargo, está cargando el fichero a Amazon S3 y activando el procesado del vídeo. El porqué de esta decisión viene definido por el Backend y el tipo de instancias elegidas para el procesado del vídeo, lo cual se explica con detalle en el apartado donde se detalla la configuración del servicio Elastic Beanstalk.

5.1.6 Códigos QR

Para lograr la correcta funcionalidad de la comprobación de asistencia por medio de códigos QR y su escaneo, la aplicación toma dos caminos diferentes, uno para la implementación del mostrado del código y otra para el escaneo de este.

Mostrado del código QR

El mostrado del código QR se basa en la realización de una petición a la API solicitando el mismo, para posteriormente mostrarlo hasta que pasa un determinado tiempo. Una vez este tiempo pasa, se vuelve a realizar la operación hasta que el usuario cancela el proceso.

Para la implementación de esta funcionalidad, el usuario ha de activar la opción de comenzar la comprobación de asistencia mediante códigos QR. Una vez realizado este paso, la aplicación automáticamente pedirá el código a través del servicio correspondiente para posteriormente mostrarlo a la vez que se activa un temporizador establecido en cinco segundos. Tras pasar este tiempo, se repite la operación.

```
void startTimer() {
  _startTimerShouldStart = false;
  const oneSec = const Duration(seconds: 1);
  _timer = new Timer.periodic(
    oneSec,
    (Timer timer) => setState(
      () {
        if (_start < 1){
          _timer.cancel();
          _start = 5;
          _startTimerShouldStart = true;
          _qrGeneratorBloc.add(QRCheckAssistanceEvent(eventId: widget.eventId));
        }
        else _start = _start - 1;
      },
    ),
  ); // Timer.periodic
}
```

Ilustración 17: Temporizador del mostrado de código QR

El mostrado del código QR se realiza mediante un módulo llamado “qr_flutter”, el cual provee un widget al que se le pasan datos que codificar, así como otros parámetros como el tamaño.

Escaneo del código QR

La implementación de la lectura del código se basa en la utilización de un módulo llamado “barcode_scan”, el cual ofrece una implementación de cámara que reconoce los patrones de las diferentes versiones de códigos QR, así como los códigos de barras. El módulo trae consigo la función asíncrona “scan”, que activa la cámara y comienza a buscar patrones.

```
void main() async {
  var result = await BarcodeScanner.scan();
  if (result.type == ResultType.Barcode &&
      result.format == BarcodeFormat.qr) {
    _qrScanBloc.add(QrScanSubmitScannedCodeEvent(uuid: result.rawContent));
  }
}
```

Ilustración 18: QR Code Scan

Tras el correcto escaneo del código por parte de la función, se añade un evento al BLoC asociado a la vista para que se comunique con el Backend, indicando los datos del evento del cual se ha escaneado el código y la marca de tiempo en el que fue realizado.

Si el servidor responde con un código de respuesta satisfactorio, significa que el usuario ha sido añadido al evento. En caso contrario, el usuario deberá de volver a escanear el código para adherirse a este.

5.1.7 Diseño final

En el siguiente apartado se muestra el resultado de la implementación de la aplicación en cuanto a diseño se refiere, el cual ha sufrido un notable cambio desde la maquetación debido, principalmente, a las ideas que iban surgiendo conforme se realizaba la implementación y se iba aprendiendo cada vez más acerca de las posibilidades que ofrecen los Widgets.

Inicio de sesión

El diseño de la vista de inicio de sesión es claro y sencillo, el usuario no debería de tener ningún tipo de duda con lo que ha de hacer en esta vista. Esta vista sirve para iniciar sesión rellenando los datos y posteriormente pulsando el botón habilitado, o para navegar a la pantalla de registro pulsando en el texto destacado para el registro.

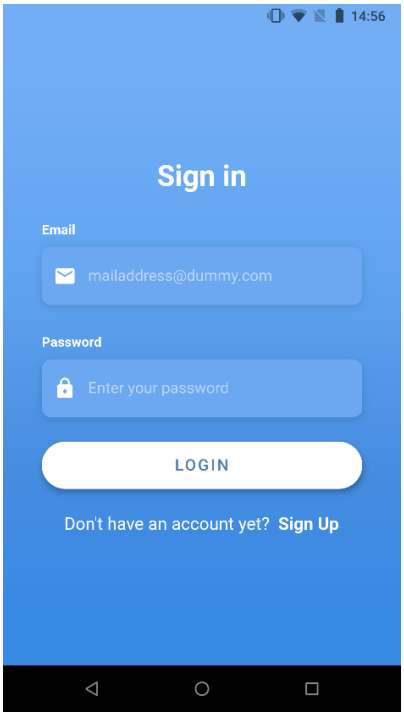


Ilustración 19: Inicio de sesión

Registro

La vista de registro consiste en un botón destacable y en una serie de campos a rellenar; Correo electrónico, contraseña, confirmación de contraseña, nombre y apellidos.

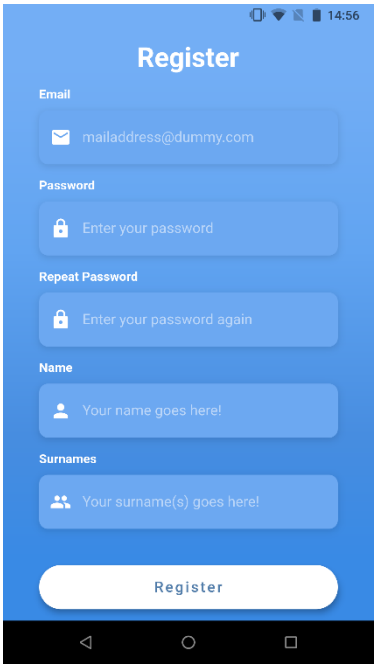


Ilustración 21: Registro

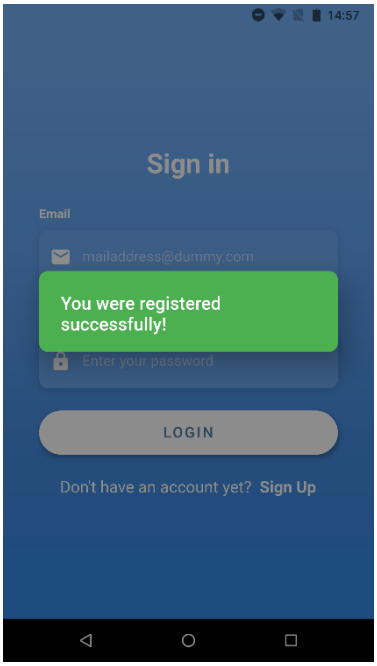


Ilustración 20: Registro confirmado

Listado de eventos

La vista que contiene el listado de eventos para el usuario consiste en un botón para refrescar el listado, el propio listado en forma de tarjetas que se pueden pulsar para navegar hacia el detalle del evento y un botón flotante con un signo "+" que sirve para añadir un evento.

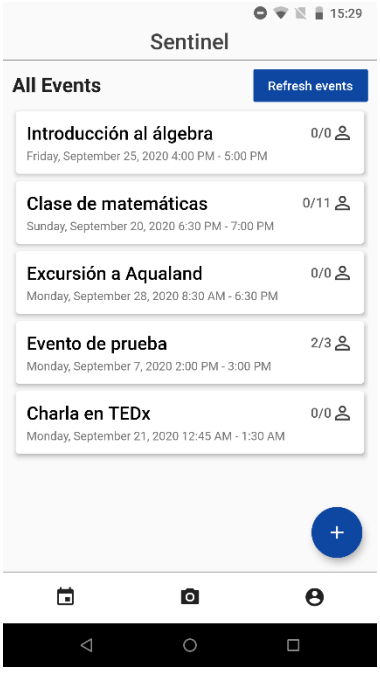


Ilustración 22: Listado de eventos

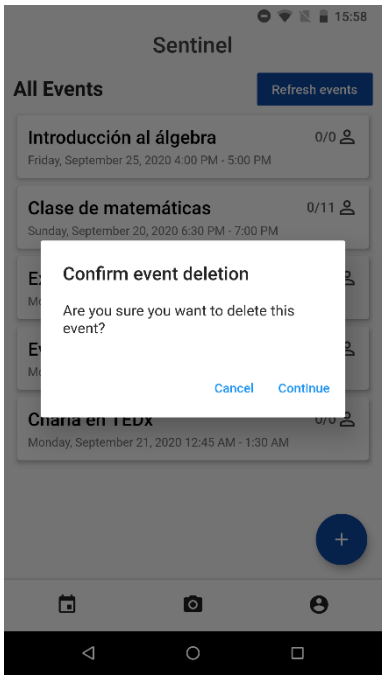


Ilustración 23: Borrar evento

Crear evento

La vista de creación de evento consiste en un botón de confirmación y una serie de campos rellenables, incluyendo campos de texto, así como selectores de fechas.

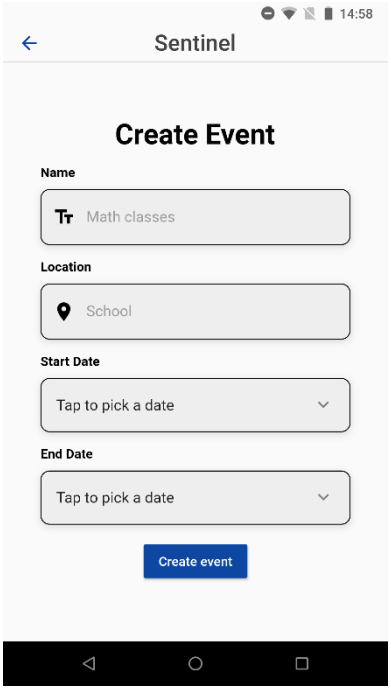


Ilustración 25: Crear evento

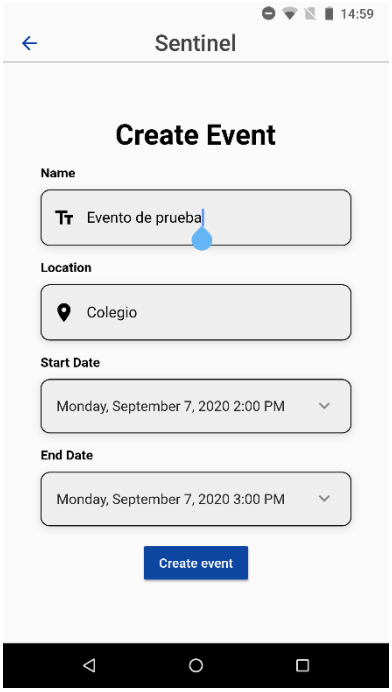


Ilustración 24: Crear evento (con datos)

Detalles de evento

La pantalla de detalles de un evento es la que contiene la información completa del mismo y da acceso a las diferentes funcionalidades relativas a este. De esta forma, se incluye la información de nombre, lugar y fechas del evento. Así como se da acceso a la lista de asistentes, a obtener la información por correo y a comenzar una comprobación de asistencia.

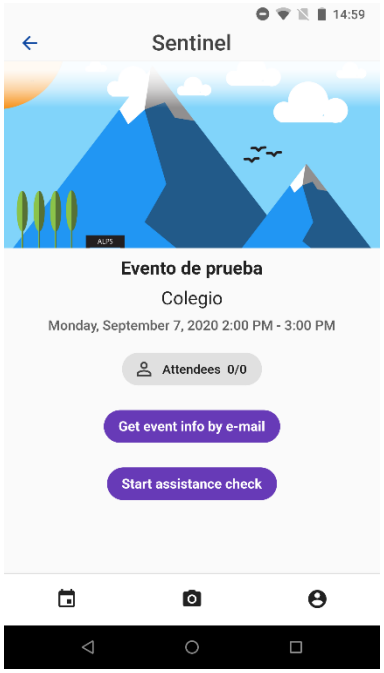


Ilustración 26: Detalles del evento

Lista de asistentes

La vista de lista de asistentes contiene, como su mismo nombre indica, una lista de los asistentes para el evento del cual estamos consultando la lista. En esta vista se puede deslizar sobre un registro de usuario hacia la izquierda para eliminarlo. Por otro lado, se incluye un botón flotante para añadir un asistente.

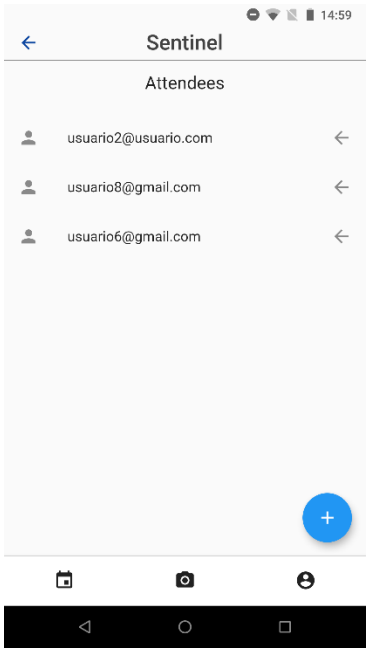


Ilustración 27: Lista de asistentes

Añadir asistente

La pantalla de lista de asistentes consiste en un campo de texto donde introducir el nombre de usuario (correo electrónico) y según el valor introducido, se mostrará una serie de usuarios que se podrán añadir al evento.

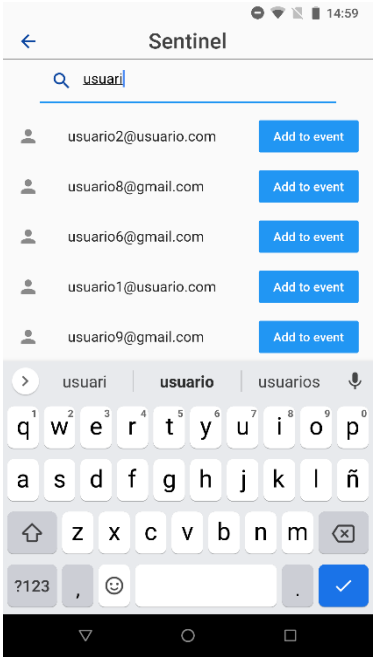


Ilustración 28: Añadir asistente

Envío de e-mail

El envío de correos electrónicos cuyo botón se muestra en los detalles del evento, muestra un mensaje de confirmación.

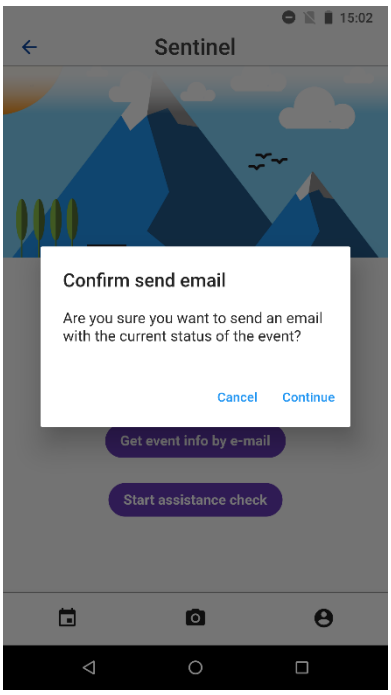


Ilustración 29: Confirmar envío de e-mail

Comenzar comprobación de asistencia

Al pulsar en el botón del detalle del evento para comenzar la comprobación de asistencia, se muestra un pequeño menú donde elegir el tipo de comprobación a realizar.

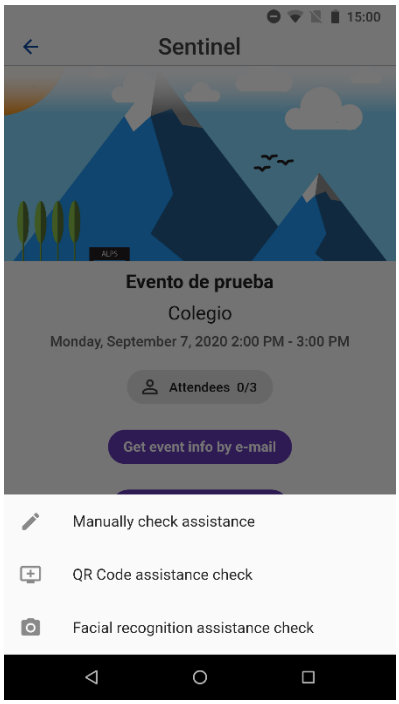


Ilustración 30: Comenzar comprobación de asistencia

Comprobación manual de asistencia

En la vista de comprobación manual de asistencia, se muestra una lista de los asistentes del evento junto con un componente “checkbox” a la derecha de cada registro de usuario. A su vez, si el asistente está confirmado, se muestra en verde, si no, en rojo.

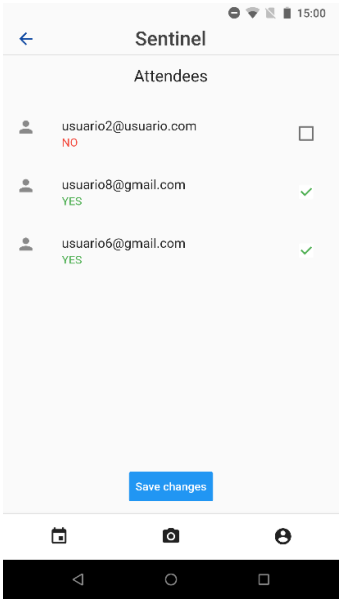


Ilustración 31: Comprobación manual de asistencia

Comprobación de asistencia por código QR (Muestra)

Esta vista corresponde a la muestra de los diferentes códigos QR generados por la aplicación para el proceso de comprobación de asistencia por estos. En la vista se muestra un temporizador para que el código sea actualizado, seguido de un predominante código mostrado. Por último, encontramos el botón para cancelar el proceso y volver a la pantalla anterior.

Ilustración 32: Muestra de código QR

Comprobación de asistencia por código QR (escaneo)

Esta vista muestra un mensaje indicando que el sistema está esperando por un escaneo de código QR. El botón habilitado para abrir el escáner realiza la acción que el propio nombre indica.

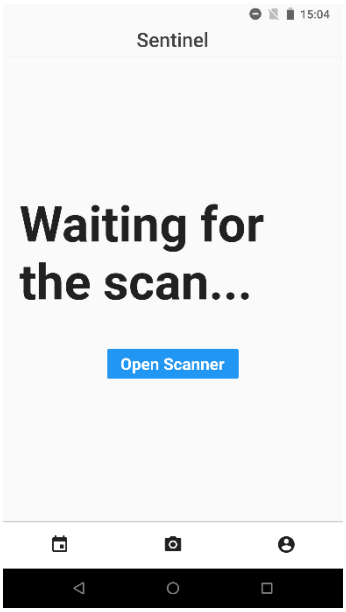


Ilustración 34: Escanear código QR

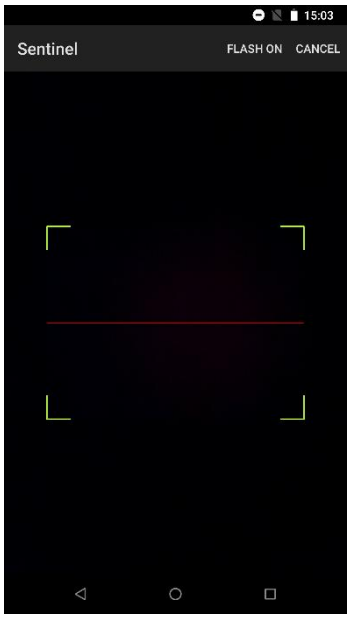


Ilustración 33: escáner

Comprobación por reconocimiento facial

La vista para la comprobación mediante reconocimiento facial incluye unas instrucciones de cómo utilizar la funcionalidad y un botón que realiza la acción que su mismo nombre indica.

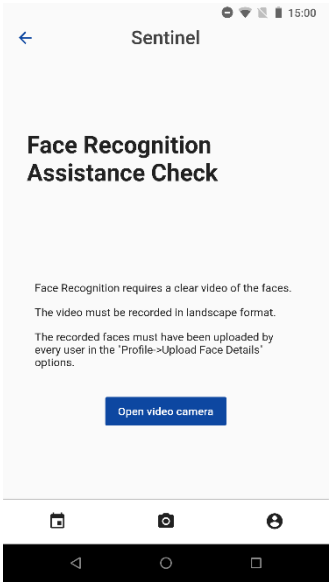


Ilustración 35: Reconocimiento facial

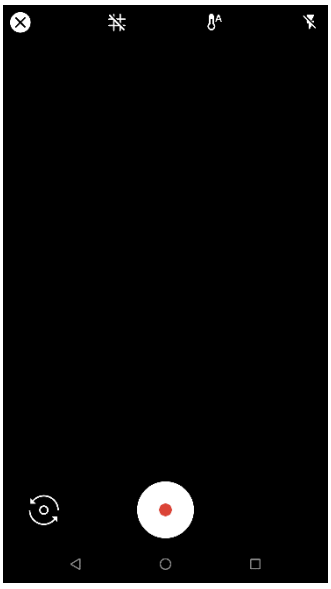


Ilustración 36: Cámara de vídeo

Perfil

La vista del perfil incluye información sobre el usuario que tiene sesión iniciada; nombre de usuario, nombre y apellidos. Además, se da una breve explicación para el uso de la indexación de rostros.

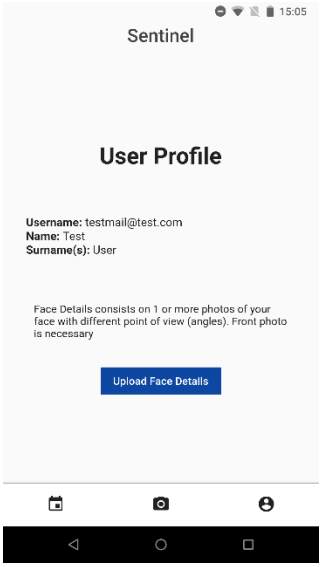


Ilustración 37: Perfil

Vista de Carga

En este apartado de diseño, también se incluye una vista sobre la carga en la aplicación. Esta vista define un estado de espera para el usuario, indicándole que un proceso, ya sea interno o de red, está sucediendo.

La vista de carga es la misma en cualquier parte de la aplicación.

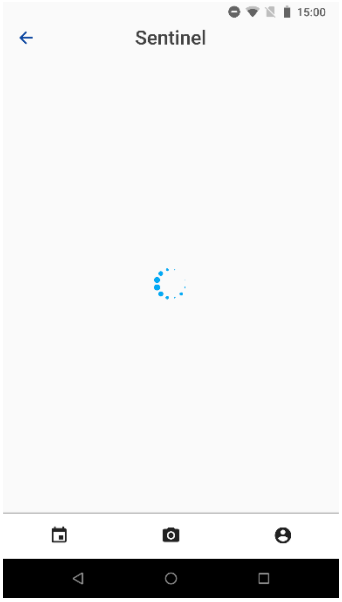


Ilustración 38: Vista de carga

5.2 Implementación del Backend

En esta sección se van a explicar las diferentes configuraciones y procesos por los que se ha tenido que pasar para desplegar los servicios en Amazon Web Services, así como las dificultades por las que se ha atravesado para la implementación de algunas funcionalidades.

5.2.1 Servicios

Como ya se ha explicado en la sección de Diseño, la arquitectura para la parte Backend o servidora de Sentinel se basa en la configuración y despliegue de microservicios provistos por la plataforma de servicios en la nube Amazon Web Services.

Para la configuración y despliegue de los servicios, hemos utilizado la conocida como "consola" de Amazon, que es una interfaz gráfica accesible mediante web para el acceso a todos los servicios que se ofrecen.

La gran mayoría de estos servicios se han ido configurando y desplegando según el caso de uso que se estuviese implementando en cada momento. Además, cada funcionalidad ha requerido de más de un servicio o parte de este, como puede ser una llamada a API Gateway que ejecuta una función Lambda y guarda un dato en DynamoDB.

Todos los servicios han sido desplegados en la zona Frankfurt.

API Gateway

API Gateway es el servicio que se encarga de preparar las direcciones y las rutas a las cuales las diferentes aplicaciones pueden realizar llamadas y este servicio las maneja dependiendo de cómo hayan sido realizadas. Normalmente podemos comparar este servicio con el controlador común en una API, la cual se encarga de recibir peticiones para redirigirlas a la capa de servicio que posteriormente responderá para devolver un resultado al usuario.

Para comenzar a utilizar este servicio hay que acceder al mismo y crear una API, en nuestro caso de tipo REST, que, tras darle un nombre, está operativa.

La creación de las diferentes rutas es trivial, tan solo basta con situarse en la ruta sobre la que queremos construir el nuevo recurso, para posteriormente con una serie de pasos e introduciendo el nombre del recurso, se crea.

Ilustración 39: Creación de un recurso con API Gateway

Tras la creación de un recurso, procedemos a crear uno o más métodos de los diferentes tipos de llamadas que se pueden realizar, para lo cual hay que indicar qué tipo de integración se desea, que en todos los casos de Sentinel ha sido una función Lambda utilizando la integración proxy que permite API Gateway a esta, que no es más que pasar todos los datos de la llamada a la función Lambda.

Además, las llamadas a una ruta pueden estar aseguradas por diferentes medios, ya sea con roles de Amazon IAM, claves de API o autorizadores de Cognito. Este último método es el utilizado en Sentinel y se realiza creando un autorizador desde la propia API apuntado al servicio Cognito desplegado para la aplicación.

En la siguiente imagen se detalla el proceso que toma una petición desde ser lanzada por el cliente hasta que vuelve al mismo con el resultado de la operación.

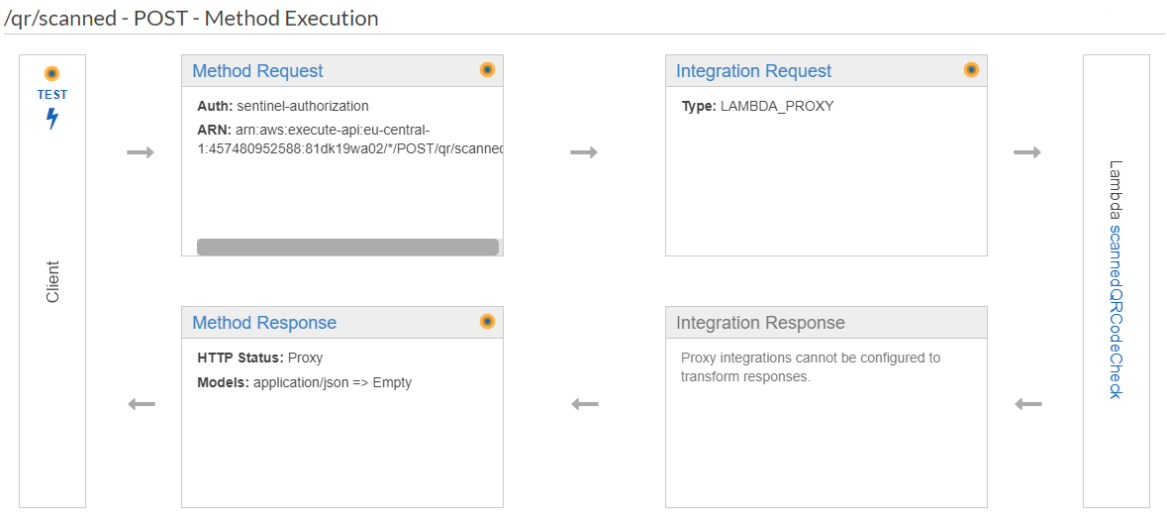


Ilustración 40: Método POST con integración Lambda

Amazon Lambda

Este servicio es el corazón del Backend de Sentinel, el cual realiza toda la lógica necesaria e interconecta la gran mayoría de servicios. Para la configuración de cada una de las funciones se ha tenido que crear la misma y seleccionar el tiempo de ejecución en el que se quiere que funcione la función Lambda. Todas las funciones Lambda de Sentinel están desarrolladas utilizando Node.js.

Para la correcta implementación de una función Lambda, esta ha de tener los permisos justos para que pueda utilizar los servicios adecuados, cumpliendo así con el Principio de mínimo Privilegio.

Una vez creada la función Lambda con la correcta configuración, se procede a implementar el código, el cual puede ser escrito en el editor que proporciona la interfaz gráfica de Amazon o cargada como archivo comprimido (dependencias incluidas). Al realizar esta acción, se procede a guardar la función, que será utilizable por los servicios que la requieran.

Amazon Simple Storage Service (S3)

El servicio de Amazon S3 ha sido utilizado para guardar contenido relacionado con la detección facial. Tal y como se explica en la sección de Diseño, en este servicio hemos utilizado tres contenedores diferentes para albergar imágenes para el indexado de caras, imágenes tras el procesado de vídeos y un último para estos.

Para crear los diferentes contenedores hemos utilizado la interfaz gráfica. Los dos contenedores que almacenan imágenes tienen configurados eventos que disparan funciones Lambda para el procesado de las imágenes que ahí se guardan.

Amazon Cognito

Para la implementación de Amazon Cognito hemos tenido que crear una piscina de usuarios (User Pools), donde se puede establecer una configuración para aplicar a los diferentes usuarios que se den de alta en ella.

En la creación de la piscina de usuarios se han debido tener en cuenta detalles como qué utilizarán los usuarios como nombre de usuario, los atributos del usuario que queremos guardar durante el registro y qué política de contraseñas se van a aplicar.

En concreto, los atributos requeridos han sido; e-mail, nombre, apellidos. El primero de estos es el que será utilizado para iniciar sesión, pues ahorramos de esta forma al usuario a recordar un atributo específico para el nombre de usuario. La política de contraseñas utilizada ha sido la de utilizar un mínimo número de 8 caractéres, una mayúscula, una minúscula y un número.

Amazon IAM

Este es un servicio que se relaciona con la seguridad en el uso de la plataforma. La configuración de este servicio ha venido en forma de creación de diferentes roles para garantizar el Principio de mínimo Privilegio. Se han creado roles para las funciones lambda y su acceso a otros servicios, así como un usuario específico para la carga de ficheros en S3.

Ilustración 41: Roles de IAM

Amazon Simple Email Service (SES)

El servicio Amazon SES permite el envío de correos a diferentes destinatarios. El caso de uso de este servicio es el envío del resumen de eventos por e-mail.

Para el envío, se ha tenido que desarrollar una plantilla en formato HTML para que, posteriormente e indicándole los datos variables, se renderice para el receptor del correo.

Para utilizar este servicio, tras una llamada a una ruta de API Gateway, se dispara una función Lambda que organiza los datos e invoca a Amazon SES, indicando el envío de un email a un receptor concreto, con unos datos y una plantilla específica.

Amazon Rekognition

Este servicio es el utilizado para el análisis de las imágenes en el proceso de comprobación de asistencia por reconocimiento facial.

En primer lugar, requiere de una colección o repositorio donde pueda almacenar las diferentes características de los rostros que se dan de alta en la aplicación. Para este paso se utiliza el programa de línea de comando de Amazon, donde se permite el manejo de mucho de los servicios a través de la terminal de comandos del dispositivo donde esté instalado.

```
PS C:\Users\timon> aws rekognition create-collection --collection-id "assistance-check"
{
    "StatusCode": 200,
    "CollectionArn": "aws:rekognition:eu-central-1:457480952588:collection/assistance-check",
    "FaceModelVersion": "4.0"
}
```

Ilustración 42: Creación de la colección de rostros

Tras la creación de la colección de rostros, Amazon Rekognition puede utilizarla para almacenar los nuevos rostros que se den de alta en Sentinel. Esta acción es invocada desde una función Lambda que se activa tras un evento de carga de objetos en el contenedor para este propósito en Amazon S3. Como se puede apreciar en la siguiente imagen, se invoca a Amazon Rekognition y se le pasa la referencia de la imagen, el identificador de la imagen y la colección donde queremos guardar la imagen.

```
async function addImageToCollection(pictureId, s3Filename) {
    let params = {
        CollectionId: 'assistance-check',
        ExternalImageId: pictureId,
        Image: {
            S3Object: {
                Bucket: 'assistance-check-index-faces',
                Name: s3Filename
            }
        }
    }
    await rekognition.indexFaces(params, param2: function (err, data) {
        if (err) console.log(err, err.stack);
        else console.log(data);
    }).promise();
}
```

Ilustración 43: Método "addImageToCollection"

Además de guardar las características de los rostros, este servicio se usa para el análisis de las imágenes y comprobar la asistencia de los asistentes a un evento. Es por esto por lo que se requiere otra implementación utilizando este servicio.

Amazon Rekognition es invocado por una función Lambda, la cual se dispara cuando una imagen de un rostro es cargada en el contenedor "assistance-check-images" de Amazon S3. Esta imagen, resultado del procesado de un vídeo, se envía al servicio para que la analiza contrastando las características del rostro de la imagen con las presentes en la colección.

Elastic Beanstalk

Este servicio surge dada la necesidad de una máquina que actúe como servidora para el procesado de los vídeos grabados para la comprobación de asistencia por reconocimiento facial. Esto es así dado el tamaño de las dependencias utilizadas para este propósito, que exceden los límites permitidos por Amazon Lambda e impiden el uso de este servicio.

El servicio Amazon Elastic Beanstalk permite la implementación de una API sin configurar ningún aspecto del servidor. En nuestro caso, dado que las dependencias utilizadas son de Python y la necesidad de una herramienta sencilla de aprender, hemos optado por utilizar el framework Flask para la creación de una sencilla API que, tras ser invocada se descargará el vídeo de Amazon S3, lo procesará y extraerá imágenes del vídeo para cargarlas en otro contenedor donde se

disparará un evento. Esta API ha sido encapsulada en un contenedor con Docker, para que su despliegue sea trivial tanto en local como el automático realizado por el servicio.

Elastic Beanstalk utiliza automáticamente el servicio Amazon Elastic Compute Cloud, que permite el despliegue de máquinas virtuales sobre infraestructura optimizada y con coste por hora. El servicio Elastic Beanstalk es gratuito, solo se paga por los demás que están relacionados con él, es decir, por las máquinas virtuales utilizadas.

Una vez desarrollado y siendo funcional el contenedor Docker con la API en Flask preparada para aceptar peticiones, se crea un entorno de servidor web en Elastic Beanstalk, donde se indica qué tipo de plataforma se va a usar (Docker), así como se requiere de la carga del archivo comprimido inicial con el código de la aplicación o de la referencia del contenedor de Amazon S3 donde se encuentra alojado. Si se usa la primera opción, el servicio crea automáticamente un contenedor para almacenar el código.

Una vez creado el entorno, Elastic Beanstalk automáticamente intenta desplegar el contenedor Docker.

En la configuración del servicio, se permite seleccionar que tipo de instancia de máquina virtual de Amazon Elastic Compute Cloud se quiere utilizar para la aplicación, estando disponible también una opción de auto escalado, la cual no utilizamos. La máquina que seleccionamos en un principio fue una instancia “t2.micro”, la cual, tras horas de problemas en el despliegue, resultó ser insuficiente para albergar la aplicación en Flask, por lo que hubo que aumentar la instancia a una “t2.small”, de un coste de 0,023 dólares americanos por hora. Esta última instancia permitió un despliegue de la aplicación en pocos minutos.

Reconocimiento Facial

La implementación de la funcionalidad de reconocimiento facial, en la parte del Backend, ha sido sin duda la más complicada del proyecto para lograr una optimización de los costes de esta funcionalidad.

Para esta funcionalidad utilizamos el servicio Amazon Rekognition, servicio de pago por demanda y el coste se especifica en dólares americanos con el símbolo $.

Amazon Rekognition permite el análisis de vídeo alojado en Amazon S3 para su procesado, siendo su coste de 0.10$ por minuto de vídeo procesado. Si Sentinel procesa 1000 vídeos de una duración aproximada de un minuto por vídeo, tenemos un coste aproximado de 100$ por cada 1000 vídeos.

En cambio, Amazon Rekognition permite también el análisis de imágenes alojadas en Amazon S3, siendo el coste de estas de 0.001$ por imagen. Si cada evento analizado tiene un promedio de 30 personas, y analizamos 1000 vídeos, tendremos un coste de 30$.

El método de análisis de imágenes parte igualmente de un vídeo producido desde la aplicación por el usuario. Es por esto por lo que debe de existir un proceso para producir las imágenes que después se cargaran a un contenedor en Amazon S3 donde se disparará automáticamente un evento para que una función Lambda envié las imágenes a analizar a Amazon Rekognition y actúe en consecuencia.

Hay un detalle muy para tener en cuenta en los cálculos realizados: las imágenes de rostros extraídas de un mismo vídeo no deben repetirse, pues mandarán a analizar un mismo rostro, costando este tantas veces más como repeticiones de este se hayan sucedido.

Por todo lo explicado, la implementación de esta funcionalidad fue tomando diferentes iteraciones hasta formar los algoritmos explicados en el apartado de Diseño.

Las iteraciones para la posterior satisfactoria implementación de la funcionalidad han sido varias.
En primer lugar, se implementó con OpenCV un programa para la extracción de imágenes del vídeo, donde de cada fotograma, se extraía una imagen. Este método resultó en una cantidad demasiado elevada de imágenes, una por cada fotograma del vídeo. Es por esto por lo que se procedió a extraer cada 15 fotogramas, de manera que la cantidad total de fotogramas era mucho menor.

Sin embargo, igualmente muchos fotogramas podrían no incluir un rostro y ser enviadas a análisis a Amazon S3, por lo que se optó por incluir una inteligencia artificial que detectase si en una imagen existían rostros o no. En caso positivo, se recogían la posición de los rostros sobre la imagen, los cuales eran recortados y extraídos a una imagen. De esta forma, nuestras imágenes siempre contendrían un rostro y tendrían un tamaño muy pequeño al estas ser recortadas. Para este propósito, se añade la librería "face_recognition", la cual ofrece muchas funcionalidades relacionadas con la inteligencia artificial y el manejo de detección de rostros.

El último paso para la actual implementación de la extracción de rostros viene dado por la necesidad de reducir aún más la cantidad de imágenes, ya que pese a tener un tamaño muy pequeño, una imagen de rostro podría repetirse multitud de veces, al aparecer en diferentes fotogramas del vídeo, mandando a analizar muchas imágenes de una misma persona. Es por esto, que se hace uso de la librería anteriormente descrita para, además de detectar las imágenes de los rostros, analizar las características de estos y almacenarlos en una variable en memoria. Este proceso de análisis nos permite contrastar cada fragmento de fotograma que contenga un rostro con los ya analizados por la aplicación, permitiendo desecharlo en caso de repetición.

El último paso en el procesado de imágenes es la carga de estas en el contenedor adecuado en Amazon S3, y el posterior borrado de las mismas del almacenamiento donde se ejecutó el procesado.

Tras la carga de imágenes, se dispara un evento que indica a una función Lambda que ha dispararse para analizar cada una estas y asignar el usuario al evento, si corresponde. Este último proceso se realiza de forma paralela y en menos de un segundo.

```python
def process_video(file):
    video_capture = cv2.VideoCapture(file)
    filenames = []
    known_face_encodings = []

    frame_count = 0
    while True:
        ret, frame = video_capture.read()
        if not ret:
            break
        frame_count += 1
        if frame_count % 15 != 0:
            continue
        # Resize frame of video to 1/4 size for faster face recognition processing
        small_frame = cv2.resize(frame, (0, 0), fx=0.10, fy=0.10)

        # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
        rgb_small_frame = small_frame[:, :, ::-1]

        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
        for face_encoding in face_encodings:
            matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
            if True in matches:
                print("Face in " + str(frame_count) + " already regonized... skipping")
                continue
            else:
                for top, right, bottom, left in face_locations:
                    cropped_frame = small_frame[top:bottom, left:right]
                    file_name = file + str(frame_count) + str(uuid.uuid4()) + '.png'
                    cv2.imwrite(file_name, cropped_frame)
                    filenames.append(file_name)
                    known_face_encodings.append(face_encoding)
    video_capture.release()
    return filenames
```

Ilustración 44: Función para el procesado de vídeo

5.3 Pruebas y validaciones

Las diferentes pruebas y validaciones realizadas durante el desarrollo del proyecto se han realizado de forma simultánea al desarrollo, de forma que conforme se implementaba una funcionalidad, esta era comprobaba durante el proceso para evitar considerar terminada una funcionalidad que no funcione de forma correcta.

La implementación se divide en dos subproyectos, las parte denominadas como Frontend y Backend, que no son más que la aplicación y la parte servidor, respectivamente. Cada una de estas partes a sido desarrollada de forma simultánea, pero considerándose independientes. Es por esto por lo que las pruebas se han dividido en las pruebas para cada una de las partes y pruebas conjuntas.

Para realizar las pruebas aisladas de la aplicación móvil, se ha utilizado el emulador de Android que viene con la herramienta utilizada para el desarrollo, Android Studio. De esta forma, cada vez que se requiere, se puede comprobar en tiempo real los efectos de los cambios realizados en el código en la aplicación, de forma que el desarrollador obtiene información para poder seguir el desarrollo de forma satisfactoria. Del mismo modo, para poder comprobar las funcionalidades que requieren de consultas externas a la aplicación se realizan peticiones ficticias, simulando la respuesta con una por el desarrollador, por ejemplo, introduciendo un JSON ficticio en una variable respuesta.

Para realizar las pruebas y validaciones para la parte servidor del proyecto, se ha realizado principalmente mediante dos métodos: El uso de la herramienta Postman para realizar peticiones a la API y el uso de la consola gráfica de Amazon Web Services para manejar los diferentes servicios de Amazon.

Si una implementación se inicia por medio de una llamada a la API, se prueba el funcionamiento de esta realizando las peticiones adecuadas mediante el uso de Postman, de forma que

podemos activar la rutina en la parte servidor, donde se dispararan en evento que activan funciones Lambda las cuales están configuradas para escribir registros de lo sucedido durante su ejecución en el servicio de Amazon CloudWatch, de forma que, si algo falla, podemos saber que es para corregirlo.

En cambio, si la funcionalidad a probar se inicia mediante, por ejemplo, la carga de un fichero en Amazon S3, las pruebas se realizan utilizando la consola gráfica, donde se carga un archivo en el contenedor adecuado para activar a través de un evento la función Lambda que lo maneje, escribiendo registros de lo sucedido.

Por otro lado, para probar la integración de la aplicación con la parte servidora, se ha realizado de dos formas distintas. En un primer lugar, se inicia el emulador de Android para comprobar las diferentes funcionalidades realizando llamadas reales a la parte servidora del proyecto y comprobando que se realizan las acciones esperadas, lo cual es indicativo de una correcta respuesta del servidor y un manejo satisfactorio de la misma por parte de la aplicación. Sin embargo, en el caso de querer probar funcionalidades que requieren del uso de la cámara de un dispositivo móvil para grabar un rostro posteriormente detectable, se ha de generar el paquete de la aplicación para un dispositivo Android real, donde probar estas funcionalidades. En el caso del proyecto Sentinel, se utilizó un dispositivo Nexus 6P.

6. Consideraciones legales

En la Unión Europea implantó en el año 2018 el Reglamento General de Protección de Datos (RGPD), relativo a la protección de las personas físicas en lo relacionado al tratamiento de sus datos personales y a la circulación de estos. Este reglamento contempla que el rostro de una persona es un dato personal por ser información que la permite diferenciarse de otra de forma unívoca.

Es por esto por lo que, para poder aplicar reconocimiento facial en una aplicación, se deben de cumplir una serie de requisitos técnicos y legales [18]:

- El usuario debe de tener una información previa y haber dado su consentimiento expreso a que la aplicación, Sentinel en nuestro caso, pueda acceder a los datos requeridos.
- La información dada al usuario debe de incluir el aviso de que se van a recoger los datos de su cara, por y para qué entidades o procesos.
- Se debe de informar al usuario del plazo de tiempo en el cual se van a conservar los datos.
- Se le debe de garantizar el derecho de acceso a los datos y al borrado de estos por parte de la empresa si hay motivo.
- Debe de existir un encargado por parte de la empresa responsable de los datos para que garantice el cumplimiento del RGPD.
- Se debe de realizar un análisis de impacto en caso de que los datos se vean comprometidos.
- Se deben de garantizar las medidas técnicas suficientes para garantizar un nivel de seguridad adecuado según el análisis de impacto.

Los datos personales recabados por Sentinel son enviados a los servidores de Amazon de forma privada y sin que estos tengan acceso a ellos. Del mismo modo, el envío se realiza por medio del protocolo HTTPS, el cual encripta la información enviada para evitar su exposición a atacantes.

Amazon Web Services, por su parte, garantiza el cumplimiento del RGPD. [19]

Sentinel es una aplicación en versión no comercial y no se distribuye al público, por lo tanto, en su versión actual cuenta con el cumplimiento de aspectos técnicos definidos por el RGPD, pero no se ha implementado la comunicación al usuario del tratamiento de sus datos.

7. Trabajo futuro

Si bien Sentinel es una aplicación completamente funcional, es cierto que diferentes factores del sistema pueden mejorarse para lograr que este sea aún más eficiente y amigable con el usuario. Estas tareas futuras son:

7.1 Mejora del proceso de reconocimiento facial

En la implementación actual, el proceso de reconocimiento facial es opaco y no instantáneo para el usuario. Esto es, el vídeo generado por la aplicación se envía a un contenedor de Amazon S3 para después ser procesado por un servidor aprovisionado y generar imágenes que posteriormente son procesadas gracias a una función Lambda y eventos en el contenedor S3.

La mejora propuesta es acelerar el proceso de reconocimiento facial, intentando que este sea en tiempo real. Esto se podría obtener, manteniendo una optimización de costes, implementando las funcionalidades de extracción de rostros de un fotograma y cribado de los ya extraídos en el dispositivo del usuario, esto es, el teléfono móvil. De esta forma, el dispositivo móvil tan solo tendría que enviar una imagen de poco tamaño que en cuestión de milisegundos será procesada para obtener un resultado, que podrá ser representado en la aplicación para añadir feedback al usuario.

7.2 Serverless

Durante el desarrollo del proyecto, se tiene mucho en cuenta el modelo de arquitectura Serverless que caracteriza a Sentinel. Esta permite obtener un coste por demanda y un rendimiento óptimo gracias al uso de una infraestructura altamente optimizada. Sin embargo, se ha requerido del uso del servicio Elastic Beanstalk debido al tamaño de las dependencias utilizadas para el procesado de vídeo, que excede el tamaño permitido por las funciones Lambda, lo cual añade la necesidad de un servidor aprovisionado.

Conviene eliminar la dependencia de un servidor aprovisionado para el procesado de vídeo y extracción de imágenes de rostros, por motivos de coste, rendimiento e integración con el resto de los servicios de Amazon. Esto se podría lograr de diferentes formas, entre las cuales se pueden encontrar lograr la reducción de tamaño de las dependencias de reconocimiento facial para que puedan utilizarse en una función Lambda, el uso de dos funciones Lambda diferentes que realicen llamadas entre sí, o la explicada en el apartado de “Mejora del proceso de reconocimiento facial”.

7.3 Mejora de la capa de usuario

En Sentinel existe la funcionalidad de registro e inicio de sesión de usuarios, necesaria para el uso de la aplicación. Se propone mejorar la capa de usuario de la aplicación para permitir funcionalidades no implementadas y manejar algunos errores de usuario que se puedan dar.

De esta forma, se propone:

- Refresco del token de acceso a la API al caducar, evitando que el usuario deba reiniciar la aplicación y volver a iniciar sesión.
- Función de recordar contraseña.
- Función de ser recordado en el dispositivo, evitando tener que iniciar sesión en el futuro.
- Eliminación de la necesidad obligatoria de cuenta de usuario para algunas funciones, como escanear un código QR.

7.4 Control de los indexados de rostros

Para que un usuario sea reconocido por el proceso de reconocimiento facial, este ha de registrar las características de su rostro mediante una funcionalidad a la cual accede mediante la aplicación. Sin embargo, el usuario puede indexar su rostro tantas veces como este desee, sin que se realice un control de cuántas veces a indexado el usuario su rostro.

Es conveniente investigar acerca de cómo guarda Amazon las características de los rostros y si existe un máximo por usuario, teniendo que implementarlo manualmente en caso de que la respuesta sea negativa, para evitar así tener un exceso de características de rostros de usuario almacenados.

7.5 Implementación de pruebas o TDD

Durante el desarrollo del proyecto, las pruebas y validaciones realizadas han sido de forma manual y analizando qué comportamientos se pueden dar en una funcionalidad y cuál es el resultado esperado, para después replicarlo en la aplicación y el Backend. Para esto se ha utilizado la aplicación mediante el simulador de Android y la herramienta Postman para realizar peticiones a la API.

Si bien estas pruebas han sido suficientes para la implementación y desarrollo de la aplicación en su magnitud actual, conforme esta evolucione se requerirán de más pruebas y de la repetición de las anteriormente realizadas para asegurar un correcto funcionamiento. De este modo, se recomiendo la implementación de pruebas en el código como práctica habitual, a poder ser mediante la aplicación de TDD (Test Driven Development).

Algunas funcionalidades eran necesarias comprobarlas de forma manual, como puede ser el reconocimiento facial al tener que cargar archivos de vídeo en Amazon S3 para que se disparasen una serie de eventos.

8. Conclusión

En esta sección se dedica a dar una opinión de la conclusión a nivel de proyecto y personal, obtenidas a lo largo del desarrollo del proyecto.

A nivel de proyecto, se han cumplido con los objetivos previstos para este, desarrollando una aplicación multiplataforma que permite la creación de eventos y realizar comprobaciones de asistencia sobre estos a través de diferentes funcionalidades.

Asimismo, se ha desarrollado una parte servidor siguiendo una arquitectura Serverless que permite un óptimo uso de recursos, pagando por estos conforme se demandan, y no aprovisionándolos. Sin embargo, un problema relacionado con la capacidad de almacenamiento de las funciones Lambda de Amazon obligó a usar un servidor aprovisionado para una función específica.

A nivel personal, este proyecto me ha permitido desarrollar por primera vez una aplicación móvil, lo cual ha sido una experiencia satisfactoria pues ha dado como resultado una aplicación multiplataforma completamente funcional y con una implementación basada en varios patrones de diseño muy interesantes. Asimismo, me ha permitido seguir adquiriendo experiencia con el uso de los servicios de Amazon Web Services y en el uso de una arquitectura Serverless.

Por otro lado, este proyecto me ha supuesto tener más en cuenta las fases de análisis y diseño de la aplicación pues, pese a que fueron realizadas, conforme iba avanzando en la implementación se hacía más patente que estas fases debían haber requerido de una profundización mayor, dado que me surgían dudas como si debería de haber optado por una base de datos relacional en lugar de haber implementado una base de datos no relacional con el servicio DynamoDB.

Considero que el proyecto ha sido desarrollado de forma satisfactoria, donde he aprendido mucho acerca de la importancia de las fases de desarrollo por las que este pasa, sobre tecnologías y lenguajes que se utilizan y, en definitiva, sobre muchas partes que componen un proyecto de software.

Bibliografía

[1] 2020 Developer Survey https://insights.stackoverflow.com/survey/2020

[2] Recopilación aplicaciones de comprobación de asistencia en eventos https://www.socialtables.com/blog/event-technology/event-check-in-apps/

[3] Attendium https://attendium.com/

[4] Brightwheel https://brightwheel.zendesk.com/

[5] Guest Manager https://www.guestmanager.com/

[6] Flutter https://flutter.dev/

[7] React Native https://reactnative.dev/

[8] Amazon Web Services https://aws.amazon.com/es/

[9] Node.js https://nodejs.org/es/

[10] Git https://git-scm.com/

[11] GitHub https://github.com/

[12] Android Studio https://developer.android.com/studio

[13] WebStorm https://www.jetbrains.com/es-es/webstorm/

[14] Adobe XD https://www.adobe.com/es/products/xd.html

[15] HTTPS https://en.wikipedia.org/wiki/HTTPS

[16] Flutter BLoC https://pub.dev/packages/flutter_bloc

[17] Serverless https://en.wikipedia.org/wiki/Serverless_computing

[18] Guía RGPD AEPD https://aws.amazon.com/es/blogs/security/all-aws-services-gdpr-ready/

[19] Amazon RGPD https://aws.amazon.com/es/blogs/security/all-aws-services-gdpr-ready/

[20] Medium https://medium.com/

[21] Towards Data Science https://towardsdatascience.com/

Apéndice- Justificación de las competencias cubiertas

IS01: En el proyecto se encuentran implementadas todas las funcionalidades pensadas durante el análisis de este y con un control de errores y excepciones que hacen que, ante un comportamiento no deseado, el sistema actúe para que el usuario no vea arruinada su experiencia de uso. Asimismo, durante el desarrollo del proyecto se ha cuidado la forma en la que se escribe el software, procurando cumplir con principios de ingeniería del software, como puede ser el de responsabilidad única, y aplicando patrones de diseño que nos permitan construir un software altamente mantenible.

IS02: Durante la etapa de análisis del proyecto, se propusieron casos de uso pensando siempre en el problema que se quería solventar, así como haber realizado estos teniendo en cuenta el tiempo de desarrollo y la existencia de herramientas, servicios o librerías de terceros que ayudarían a acelerar la construcción del software. Por ejemplo, se hace uso de librerías de reconocimiento facial y diferentes servicios de Amazon.

IS03: Para un desarrollo satisfactorio de Sentinel, se han tenido que integrar diferentes patrones de diseño y arquitecturas software específicas para aprovechar al máximo las ventajas de cada tecnología. Asimismo, se han tenido que interconectar los diferentes módulos del sistema para que funcionen correctamente, aunque utilizan tecnologías de terceros distintas. Por ejemplo, se ha utilizado el patrón BLoC en el desarrollo de la aplicación para aprovechar los estados y eventos en los que se basa Flutter.