# MEMORIA TÉCNICA

## Videona_Talentum

Proyecto desarrollado para Android e iOS


Grupo de trabajo: MaskFun

Mar Fernández
Lidon Campos
Tamara Arévalo
Adrián Álvarez
Kevin Fernández

## El objetivo:

Videona pone a disposición del usuario herramientas profesionales de edición de video, accesibles desde sus dispositivos móviles. En Videona buscan añadir funcionalidad al set de herramientas de edición que tienen. Por ello nos han solicitado el siguiente objetivo.

El objetivo que nos ha pedido Videona, ha sido realizar dos aplicaciones nativas, una en Android y otra en Swift. Para poder llegar a todos los clientes que utilicen su aplicación de edición de video.

En esta aplicación las tareas que teníamos que obtener son las siguientes:

1. Poner en funcionamiento la cámara delantera del dispositivo.
    a. Cámara delantera.
    b. Cámara trasera.
2. Detectar una o varias caras.
    a. Dibujar cuadrado, para la cara y puntos para los ojos y la boca.
3. Identificar la posición de los ojos y la boca.
4. Colocar las máscaras en el lugar elegido, ojos y boca (sombrero y bigote).
5. Realizar un seguimiento de la máscara, junto con la cara.
6. Escalar las imágenes, en función de su posición. Es decir, cuanto más lejos, más grande.
7. Guardar un fichero de vídeo con la composición creada durante el proceso.
8. Compartir fichero por Whatsapp.

A partir, de los recursos que Videona nos proporcionó, empezamos nuestro trabajo.

## Cuestiones técnicas:

| | ANDROID | iOS - SWIFT |
|---|---|---|
| Iniciar la cámara. | Búsqueda de librerías, implementación y resolución de problemas derivados del uso de permisos. | Función: initialiseCaptureSession()<br>Librería: AVFoundation.<br>Clase AVCaptureSession. |
| Cambiar la posición (back, front). | Creamos una función changeCamera() asociada a un botón, que nos cambia la cámara de trasera a delantera. | Función: changeCamera()<br>Librería: AVFoundation.<br>Clase: AVCaptureDevice. |
| Detectar caras. | La propia librería de GoogleVision nos proporcionaba la detección de caras. | Librería: CoreImage<br>Clase CIFeature. |

| | | |
|---|---|---|
| Identificar la posición de los ojos y la boca. | La librería GoogleVision nos proporciona una serie de funciones que nos devuelven los distintos valores de ojos, boca, etc. | Librería: CoreImage.<br>Clase: CIFaceFeature.<br>Variables: rightEyePosition, leftEyePosition, mouthPosition. |
| Colocar las máscaras en el lugar elegido. | Lo controlamos con una función con booleanos. Que pinta en función de si está activo o no. | Función controlada con **If**, para que cada vez que el usuario seleccione una máscara se acumulen las imágenes en la foto. |
| Realizar un seguimiento de la máscara, junto con la cara. (Rotación) | La máscara se va pintando en función de la posición recogida, la rotación la realizamos mediante un parámetro que nos facilita la librería. Y realizamos matrices para generar la rotación. | Framework:<br>import CoreGraphics<br>import Darwin<br><br>Función:<br>CGAffineTransformRotate(posición imagen escalada, ángulo rotación)<br>Variable: CGFloat |
| Escalar las imágenes. | Encapsulamos la imagen dentro de un rectángulo, cuyas dimensiones están delimitadas en función de las dimensiones de la cara. | Función:<br>CGAffineTransformMakeScale(x,y)<br>Librería:<br>import CoreGraphics<br>import Darwin<br>Variable: CGFloat. |
| Multitracking. | La librería GoogleVision realiza directamente el multitracking. | FrameWork:<br>FrameWorkCoreImage()<br>Clases:<br>CIImage, CIDetector, CIFaceFeature. |
| Guardar un fichero en formato vídeo. | Hemos utilizado la librería MediaProjection para capturar el vídeo a partir de la screen . El vídeo se guarda en la galería. | Hemos conseguido grabar, pero nos queda como mejora guardarlo en el móvil. |