

# Trabajo de Fin de Grado

Grado en Ingeniería Informática

---

## Desarrollo e implementación del sistema Smart Dash Cam

*Development and implementation of the  
Smart Dash Cam system*

Juan Espinosa González

---

La Laguna, 5 de septiembre de 2017

D. **Pino Caballero Gil**, con N.I.F. 45.534.310-Z Catedrática de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora.

D. **José Iván Santos González**, con N.I.F. 78.637.989-T Investigador FPI adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor.

## C E R T I F I C A (N)

Que la presente memoria titulada:

*“Desarrollo e implementación del sistema Smart Dash Cam.”*

ha sido realizada bajo su dirección por D. **Juan Espinosa González**, con N.I.F. 78.850.684-Z.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de septiembre de 2017.

## Agradecimientos

A mis padres, por su dedicación y paciencia.

A Pino e Iván, mi tutora y cotutor respectivamente, por su apoyo  
en este proyecto.

# Licencia



© Esta obra está bajo una licencia de Creative Commons  
Reconocimiento-NoComercial-CompartirIgual 4.0  
Internacional.

## **Resumen**

En este Trabajo de Fin de Grado se presenta el sistema Smart Dash Cam cuyo fin es mejorar la asistencia a personas en accidentes de tráfico. El desarrollo incluye una aplicación móvil capaz de detectar si un usuario ha sufrido un accidente y una aplicación web para consultar de forma instantánea el estado y los datos del usuario.

Smart Dash Cam utiliza el dispositivo móvil para realizar capturas de vídeo durante la conducción y conocer cuándo se produce un accidente. En caso de producirse, se envían a un servidor los datos y ubicación del usuario y la secuencia de vídeo capturada, y se inicia una retransmisión de audio y vídeo en tiempo real desde el dispositivo. A través de la plataforma web, los servicios de emergencias u otras entidades, podrían obtener la información necesaria para actuar con mayor rapidez y disminuir así los tiempos de respuesta.

**Palabras clave:** Android, Dashcam, Aplicación móvil, Asistencia.

## **Abstract**

This Degree Final Project develops the Smart Dash Cam system to assist victims of traffic accidents. The development includes a mobile application to detect if any user has suffered an accident and a web application to consult instantly the user status and the user data.

Smart Dash Cam uses the mobile device to make video captures while driving and to know when an accident occurs. If so, the data and location of the user and the captured video sequence are sent to a server, and a real-time video and audio streaming is initiated from the device. Through the web platform, emergency services or other entities could obtain the information needed to act faster and thus reduce response times.

**Keywords:** Android, Dashcam, Mobile Application, Assistance.

# Índice General

<b>Capítulo 1. Introducción</b>	<b>1</b>
1.1 Motivación .....	1
1.2 Objetivos.....	2
1.3 Fases .....	3
1.4 Estructura de la memoria.....	4
<b>Capítulo 2. Introducción a la aplicación</b>	<b>5</b>
2.1 Definición del problema.....	5
2.2 Estado del arte .....	6
2.2.1 S.O.S Emergencias .....	6
2.2.2 AutoBoy Dash Cam .....	7
2.3 Conceptualización de la propuesta.....	8
2.4 Aspectos legales.....	8
<b>Capítulo 3. Tecnologías</b>	<b>10</b>
3.1 Aplicación móvil.....	10
3.2 Aplicación web .....	11
3.3 Tecnologías de servidor .....	12
3.3.1 Firebase .....	12
3.3.2 Wowza Streaming Engine .....	13
3.3.3 WampServer .....	13
3.4 Seguridad .....	14
3.5 Otras herramientas.....	15
<b>Capítulo 4. Desarrollo</b>	<b>16</b>
4.1 Arquitectura general del sistema .....	16
4.2 Aplicación móvil.....	17
4.2.1 Autenticación.....	20
4.2.2 Configuración.....	21

4.2.3	Auto-arranque.....	22
4.2.4	Rutina de grabación de vídeo.....	24
4.2.5	Detección de colisiones.....	27
4.2.6	Interacción por voz.....	27
4.2.7	Envío de datos.....	27
4.2.8	Streaming.....	29
4.3	Aplicación web.....	31
4.3.1	Carga y actualización de datos.....	32
4.3.2	Streaming.....	33
4.3.3	Vídeo del accidente y ubicación del usuario.....	34
4.4	Servidor.....	35
4.4.1	Configuración de Firebase.....	35
4.4.2	Configuración de Wowza Streaming Engine.....	38
4.5	Seguridad.....	38
4.5.1	Seguridad en aplicaciones.....	38
4.5.2	Seguridad en Firebase.....	38
<b>Capítulo 5.</b>	<b>Presupuesto</b>	<b>40</b>
5.1	Personal.....	40
5.2	Componentes.....	41
5.3	Coste total.....	41
<b>Capítulo 6.</b>	<b>Conclusiones y líneas futuras</b>	<b>42</b>
<b>Capítulo 7.</b>	<b>Conclusions and future Work</b>	<b>44</b>
<b>Apéndice A.</b>	<b>Interfaz de la aplicación móvil</b>	<b>46</b>
A.1.	Pantalla de autenticación / registro.....	46
A.2.	Pantalla de formulario.....	47
A.3.	Pantallas de configuración.....	48
A.4.	Pantalla principal (grabación de vídeo).....	49
<b>Apéndice B.</b>	<b>Interfaz de la aplicación web</b>	<b>50</b>



B.1. Vista de la aplicación web .....50

**Bibliografía** ..... **51**

# Índice de figuras

Figura 1.1. Logo Smart Dash Cam.....	1
Figura 2.1. Gráfica de accidentes de la DGT. ....	5
Figura 2.2. Aplicación S.O.S Emergencias.....	7
Figura 2.3. AutoBoy Dash Cam.....	7
Figura 4.1. Distribución de versiones de Android.....	17
Figura 4.2. Estructura del proyecto de la aplicación móvil.....	18
Figura 4.3. Ciclo de vida de una Activity. ....	23
Figura 4.4. Diagrama de estados de MediaRecorder.....	25
Figura 4.5. Estructura del proyecto de la aplicación web.....	31
Figura 4.6. Firebase Authentication.....	36
Figura 4.7. Firebase Realtime Database.....	37
Figura 4.8. Firebase Storage. ....	37
Figura 4.9. Reglas de seguridad en Firebase Realtime Database. ....	39
Figura 4.10. Reglas de seguridad en Firebase Storage.....	39

# Índice de tablas

Tabla 5.1. Presupuesto del personal.....	41
Tabla 5.2. Presupuesto de los componentes.....	41
Tabla 5.3. Presupuesto total.....	41



# Capítulo 1.

## Introducción

### 1.1 Motivación

Gracias a la evolución de los smartphones y a la creciente integración de la tecnología en la sociedad, cada día es mayor el número de usuarios de dispositivos móviles y, por tanto, la demanda en el mercado de aplicaciones. De este modo, teniendo la posibilidad de utilizar una tecnología al alcance de la mayor parte de la población, ¿cómo resistirse al afán de crear una nueva herramienta que sirva de utilidad a muchos?



Figura 1.1. Logo Smart Dash Cam.

Cada año miles de personas sufren accidentes de tráfico [1]. Esto lleva a una constante búsqueda de soluciones por parte de las organizaciones, y a la aparición de avances tecnológicos, con el fin de mejorar la situación de los afectados y de reducir en lo posible el número de víctimas en la carretera. Tiempo atrás, uno de estos avances fueron las dashcams [2]. Estos dispositivos, muy extendidos en países como Rusia debido a los intentos de fraude con los seguros, son cámaras que se utilizan en el salpicadero de los vehículos para obtener evidencias de lo ocurrido en accidentes de tráfico.

Smart Dash Cam se presenta como una mejora de las dashcams tradicionales. Permite utilizar los dispositivos móviles para capturar secuencias de vídeo y audio, y enviar dichas secuencias y otros datos de interés, a una plataforma centralizada, que los servicios de emergencia pueden consultar en tiempo real con el fin de optimizar las acciones de respuesta en situaciones de peligro.

¿Qué puede aportar Smart Dash Cam? Este sistema da la posibilidad al usuario final de tener en su Smartphone la funcionalidad de las dashcams a un coste reducido, además de muchas otras funcionalidades de interés explicadas a lo largo de este documento.

## 1.2 Objetivos

El objetivo principal de este proyecto es el desarrollo de una aplicación que mejore el funcionamiento de la tecnología de las dashcams, añadiendo la opción de realizar streaming con un servidor con el fin de ser de utilidad en el ámbito de la asistencia en carreteras. Para ello, se establecen los siguientes objetivos específicos:

- Diseño e implementación de la aplicación móvil Smart Dash Cam que incluye las funcionalidades de autenticación con usuario y contraseña, auto-arranque vía Bluetooth, grabación de vídeo con duración configurable, detección de colisiones, streaming, interacción por voz, cálculo de la ubicación del usuario, y envío de toda la información necesaria a un servidor en caso de accidente.
- Implementación de una plataforma web que permite consultar los datos enviados por la aplicación móvil, el contenido de los vídeos de las víctimas de accidentes de tráfico y su ubicación en Google Maps.

Además, se plantea como objetivo para el desarrollo del proyecto incluir medidas de seguridad en el acceso a los datos y en las comunicaciones entre las aplicaciones implicadas.

## 1.3 Fases

El desarrollo del proyecto se ha dividido en distintas fases que se han completado a lo largo del periodo de ejecución planteado. Las actividades a realizar, consisten en una serie de hitos acumulativos, de manera que sea más fácil observar la evolución del trabajo, así como la propia realización del mismo. A continuación se describen los hitos fijados:

- Primer hito. Estudio bibliográfico y estado del arte de dashcams y dispositivos móviles: Se realiza un estudio previo de las tecnologías actuales y se recopila toda la información necesaria en relación al proyecto con el objetivo de conocer la viabilidad del mismo.
- Segundo hito. Primer diseño de la aplicación móvil: Se establecen las tecnologías a utilizar en el proyecto y se presenta una primera aproximación a la aplicación móvil para obtener una idea de la viabilidad del mismo.
- Tercer hito. Desarrollo e implementación de aplicación móvil: Se presenta la versión final de la aplicación móvil.
- Cuarto hito. Desarrollo e implementación del servidor web: Se presenta la versión final de la aplicación web.
- Quinto hito. Implementación de mecanismos de seguridad en las comunicaciones y datos: Se exploran las distintas opciones para la implementación de seguridad en las comunicaciones y los datos. Se aplican las más viables para el proyecto.
- Sexto hito. Batería de tests para verificar el correcto funcionamiento de la aplicación: Se integran las aplicaciones que intervienen en el proyecto, se realizan pruebas de funcionamiento y se procede al análisis de los resultados y a la corrección de bugs.

## 1.4 Estructura de la memoria

A continuación se describe brevemente cada capítulo de esta memoria:

- Capítulo 1. Introducción al trabajo: En este capítulo se introducen los motivos y objetivos para el desarrollo del proyecto.
- Capítulo 2. Introducción a la aplicación: En este capítulo se realiza una presentación de la aplicación desarrollada en este proyecto, explicando sus características y funcionalidades y realizando una comparativa con otras aplicaciones existentes.
- Capítulo 3. Tecnologías: En este capítulo se expone una definición de las diferentes tecnologías utilizadas en el desarrollo del proyecto, dividiendo el contenido entre las herramientas utilizadas en la aplicación móvil y las utilizadas en la aplicación web. Además, se proponen alternativas para algunas de las tecnologías mencionadas.
- Capítulo 4. Desarrollo: En este capítulo se da una explicación detallada del sistema desarrollado en este proyecto. Se trata en profundidad el funcionamiento de las diferentes aplicaciones y tecnologías que lo integran y se explican las medidas de seguridad utilizadas.
- Capítulo 5. Presupuesto: En este capítulo se presenta un presupuesto del coste total de la elaboración del proyecto, desde los requisitos software y hardware hasta los costes de personal.
- Capítulo 6: Conclusiones y trabajos futuros: En este capítulo se exponen las conclusiones obtenidas tras la realización del proyecto y se plantean posibles modificaciones, mejoras y trabajos futuros en relación al mismo.



# Capítulo 2.

## Introducción a la aplicación

### 2.1 Definición del problema

Según la DGT [3], en 2016 se registraron 1160 fallecidos en accidentes de tráfico. En el último año, se incrementaron las cifras con respecto a 2015 en accidentes mortales, en el número de fallecidos y en heridos hospitalizados. Asimismo, en 2016 la accidentalidad en carretera fue de un total de 3,2 fallecidos diarios.

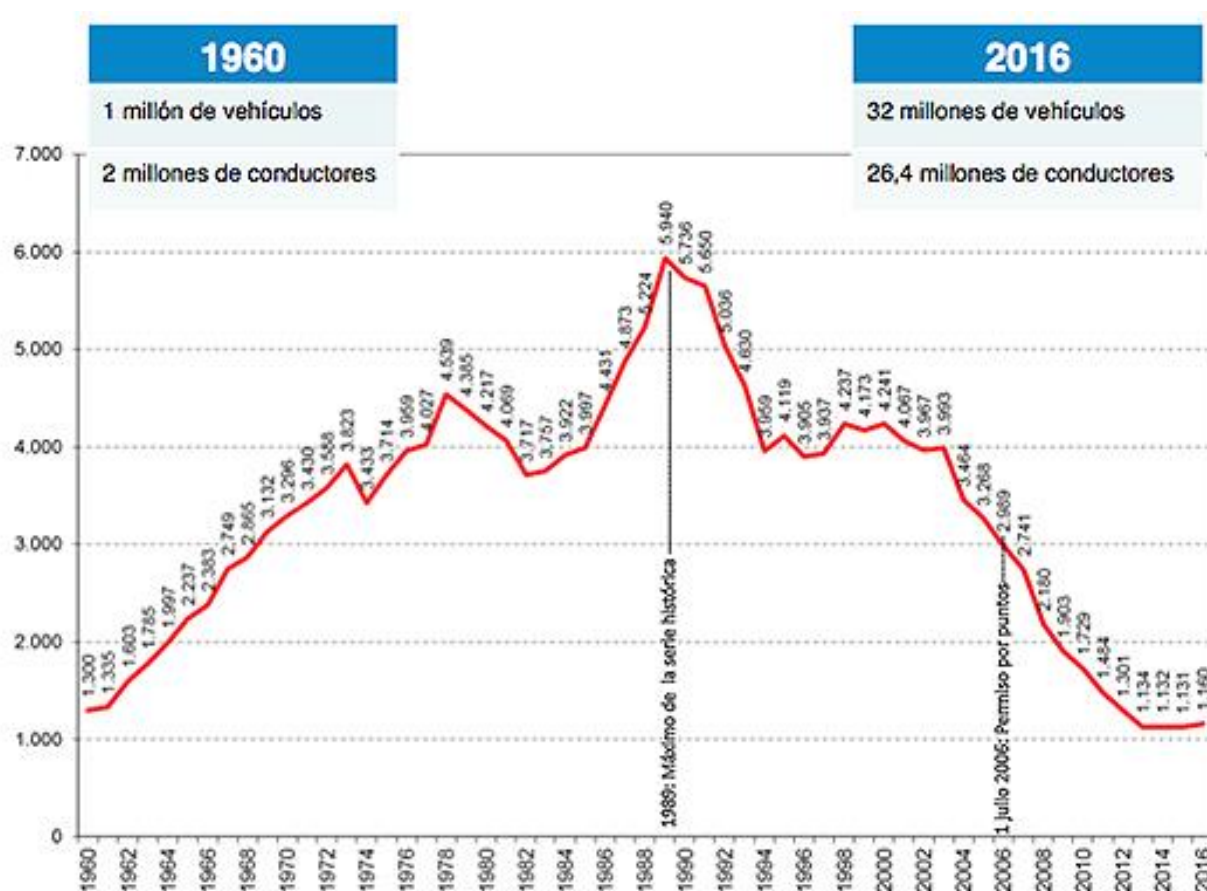


Figura 2.1. Gráfica de accidentes de la DGT.

Más de la mitad de las muertes en accidentes de tráfico se producen durante los 20 minutos posteriores al accidente. En estos casos es fundamental una buena coordinación y respuesta por parte de los servicios de emergencias para salvar la vida de los implicados.

En base a estos datos, surge la idea de desarrollar una aplicación que utilice el dispositivo móvil, una tecnología que está presente en la vida de la mayor parte de la población, para automatizar el contacto con los servicios de emergencia con el fin de salvar la vida a sus usuarios.

## **2.2 Estado del arte**

Hoy en día, en el mercado de Google Play [4], existe una gran variedad de aplicaciones de asistencia en carretera. Las aplicaciones mejor valoradas implementan múltiples opciones como el envío de la localización del usuario o el contacto con los servicios de emergencia. Asimismo, es posible encontrar aplicaciones que incluyen la funcionalidad de las dashcams. Estas aplicaciones priorizan la grabación constante de vídeo y no permiten la interacción con el usuario ante un problema.

A continuación se muestran algunos ejemplos de aplicaciones con las características descritas.

### **2.2.1 S.O.S Emergencias**

S.O.S Emergencias es una aplicación de uso exclusivo en España que ofrece la posibilidad de contactar en situaciones de emergencia con diferentes servicios como pueden ser el cuerpo de policías o bomberos. La aplicación no es específica para la asistencia en carreteras pero podría servir para ese fin. Está diseñada para personas con discapacidad auditiva o del habla y es accesible para personas con discapacidad visual haciendo uso de Google Talkback [5]. Además, la aplicación permite el envío de la ubicación del usuario.

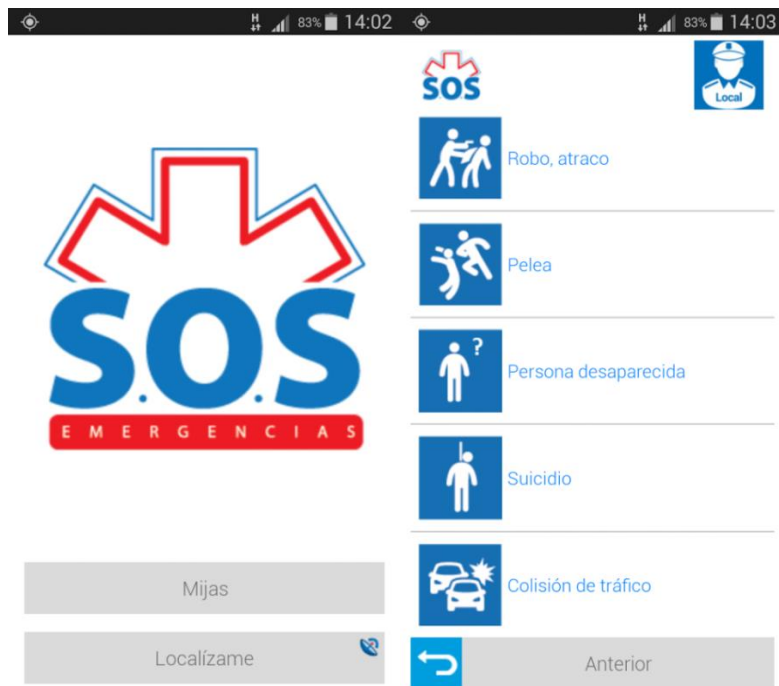


Figura 2.2. Aplicación S.O.S Emergencias.

### 2.2.2 AutoBoy Dash Cam

AutoBoy Dash Cam es una aplicación que ofrece diversas funcionalidades durante la conducción. Con un funcionamiento similar al de las dashcams y haciendo uso del dispositivo móvil, da al usuario la posibilidad de realizar grabaciones de vídeo en sus viajes, tomar fotos o utilizar la opción de GPS.

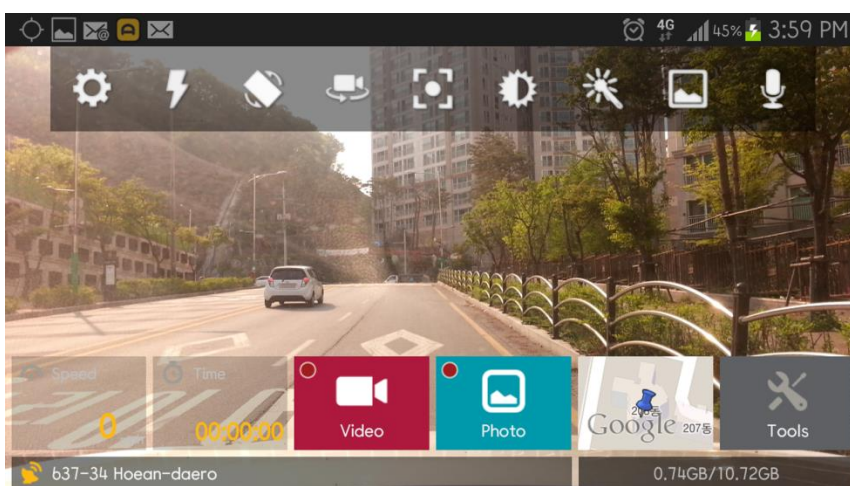


Figura 2.3. AutoBoy Dash Cam.

Es de importancia indicar que no se han encontrado ejemplos de una aplicación que integre todas las funcionalidades de Smart Dash Cam.

## **2.3 Conceptualización de la propuesta**

El sistema Smart Dash Cam pretende ser de utilidad a los servicios de emergencia y diferenciarse del resto de aplicaciones del mercado actual a través de la automatización de funcionalidades que permitan asistir al afectado en caso de accidente. Por tanto, al tratarse de una aplicación utilizada durante la conducción, se debe priorizar la seguridad de los usuarios.

Uno de los primeros pasos para lograr el objetivo propuesto es cuidar en detalle la usabilidad de la aplicación. Tras realizar la configuración de Smart Dash Cam en el dispositivo móvil, la interacción con la aplicación es mínima y está basada en respuestas por voz, lo que conlleva a una menor distracción por parte del usuario.

Una vez se inicia la aplicación se ejecuta automáticamente un bucle de grabación de vídeo de duración determinada y se lanza un algoritmo de detección de accidentes. En caso de detectarse un accidente, la aplicación realiza la única interacción posible con el usuario, que consiste en consultar si requiere de asistencia. El usuario responde por voz a la aplicación, y en caso afirmativo, se procede a enviar los datos y la ubicación del afectado y el vídeo del accidente, y se inicia un streaming en tiempo real con un servidor.

El sistema Smart Dash Cam provee a los servicios de emergencias y otras entidades oficiales de una aplicación web para visualizar todos los datos y la retransmisión de los usuarios en directo, lo que permite analizar la situación de los afectados y tomar las medidas adecuadas.

Con estas características, la aplicación espera ser de ayuda en el ámbito de la asistencia en carreteras y, mediante la tecnología, disminuir el número de víctimas de accidentes de tráfico.

## **2.4 Aspectos legales**

La legislación vigente afecta al uso de Smart Dash Cam.

Según el artículo 5.1. f) del Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de desarrollo de la Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal, se define “dato de carácter personal” como:

*“Cualquier información numérica, alfabética, gráfica, fotográfica, acústica o de cualquier otro tipo, concerniente a personas físicas identificadas o identificables”.*

Asimismo, el artículo 6.1, que habla sobre el consentimiento al tratamiento de datos personales, indica:

*“el tratamiento de imágenes de carácter personal requerirá el consentimiento inequívoco del afectado, salvo que la ley disponga otra cosa”.*

Y, por si fuera poco, la Instrucción 1/2006 de la Agencia Española de Protección de Datos (AEPD) [6], sobre el tratamiento de datos personales con fines de vigilancia a través de sistemas de cámaras o videocámaras, indica en su artículo 4:

*“las cámaras y videocámaras instaladas en espacios privados no podrán obtener imágenes de espacios públicos salvo que resulte imprescindible para la finalidad de vigilancia que se pretende, o resulte imposible evitarlo por razón de la ubicación de aquéllas”.*

Por lo tanto, teniendo en cuenta estos artículos y considerando que los vehículos se mueven por la vía pública, se podría concluir que a día de hoy hacer uso de este tipo de aplicaciones está prohibido por ley en España.

Sin embargo, recientemente la AEPD ha mostrado flexibilidad en un comentario sobre el uso de cámaras en vehículos [7]:

*“La normativa de protección de datos no es de aplicación a los ficheros mantenidos por personas físicas en el ejercicio de actividades exclusivamente personales o domésticas. En el caso planteado se podría aplicar la excepción doméstica y realizar la grabación de los viajes, siempre y cuando dicha grabación fuese para uso estrictamente personal...”*

Quizás, con algo de movilidad por parte de los servicios de emergencia y otras entidades, se pueda encontrar una excepción y conseguir llevar proyectos de este tipo a un entorno real.

# Capítulo 3.

## Tecnologías

Previamente al desarrollo, se ha realizado un estudio de investigación de las diferentes tecnologías actuales y un análisis de los requisitos del proyecto. A continuación se describen las tecnologías utilizadas en la aplicación móvil y la aplicación web, las tecnologías de servidor, las tecnologías implicadas en la implementación de medidas de seguridad y otras herramientas de utilidad en el desarrollo de Smart Dash Cam.

### 3.1 Aplicación móvil

La aplicación móvil ha sido desarrollada para el sistema operativo Android, descartando la posibilidad de realizar un desarrollo para iOS o un desarrollo multiplataforma con tecnologías como Phonegap, Cordova o Ionic.

Smart Dash Cam pretende llegar al mayor número de usuarios posibles. En lo referido a la cuota de mercado, Android consigue superar a iOS, siendo actualmente el sistema operativo más utilizado a nivel mundial [8]. Además, existen otros motivos para descartar iOS, como puede ser la necesidad de disponer de un ordenador y de un dispositivo móvil con este sistema operativo para realizar el desarrollo.

Si se busca llegar al mayor número de usuarios posibles, ¿por qué no realizar una aplicación multiplataforma? El desarrollo multiplataforma en dispositivos móviles es viable si los requisitos de la aplicación son poco exigentes con el rendimiento del dispositivo, como por ejemplo, en la construcción de aplicaciones limitadas a funcionalidades de consulta y visualización de datos, que deleguen los procesos de consumo a un servidor. Sin embargo, este tipo de aplicaciones suelen presentar problemas cuando se requiere el uso de componentes nativos de los dispositivos.

Por los motivos expuestos, se decide utilizar Android como plataforma para el desarrollo de Smart Dash Cam.

El SDK (Software Development Kit) de Android [9] está compuesto por un conjunto de herramientas de desarrollo que incluye un depurador de código, multitud de librerías, un simulador de Smartphone, documentación, ejemplos y tutoriales. Además, en el desarrollo se ha hecho uso de diversas librerías de Android, Google y de terceros, como Material Design [10] para la implementación de la interfaz de usuario, las librerías de Google Play-Services [11] para la autenticación y localización, y Libstreaming [12] para la implementación de la funcionalidad de streaming.

Por último, indicar que el IDE utilizado en el desarrollo es Android Studio y los lenguajes de programación son Java, en la parte funcional, y XML, en la interfaz gráfica.

## 3.2 Aplicación web

La tecnología utilizada en el desarrollo de la aplicación web depende en gran medida de la elección de las tecnologías de servidor.

En un principio se planteó implementar un servidor con el framework MEAN (MongoDB, Express, AngularJS, Node.js) [13].

Tras investigar las posibilidades que aporta la plataforma de desarrollo Firebase, se descartó MEAN como framework de desarrollo web y se optó por implementar una página web simple utilizando HTML, CSS, y JS.

Para la realización de esta página web se ha utilizado el editor Visual Studio Code, un editor ligero que soporta gran cantidad de tecnologías como Java, C++, HTML y JS y permite la configuración e instalación de plugins de forma sencilla.

Además, la aplicación web utiliza la librería jQuery [14], que consiste en una librería JavaScript que define funciones para facilitar la interacción entre los elementos, la gestión de los eventos y otras necesidades del desarrollo; y un framework CSS basado en Material Design llamado Materialize [15]. Además, para visualizar el streaming en directo de los usuarios utiliza el framework VideoJS [16].

## 3.3 Tecnologías de servidor

Entre las tecnologías de servidor utilizadas se incluye un servidor de autenticación, una base de datos, un servidor de streaming y un servidor web. A continuación se analiza cada una de estas tecnologías y se explican los motivos que llevaron a seleccionarlás para este proyecto.

### 3.3.1 Firebase

Firebase [17] es una plataforma de desarrollo de aplicaciones móviles y aplicaciones web adquirida por Google en 2014, que surgió con el objetivo de proporcionar una API para almacenar y sincronizar datos en la nube en tiempo real. Sus características fundamentales se pueden agrupar por los servicios que presta, los cuales se indican a continuación:

- **Analíticas.** Provee de una solución para gestionar las analíticas en las aplicaciones.
- **Desarrollo.** Aporta herramientas que facilitan la construcción de las aplicaciones, ofreciendo servicios de autenticación, bases de datos en tiempo real, almacenamiento o hosting, entre otros.
- **Crecimiento.** Permite realizar la gestión de los usuarios de las aplicaciones a través de diversas funcionalidades como pueden ser las notificaciones y las invitaciones por email o sms.
- **Monetización.** Integra Admob, que es un producto que permite obtener ganancias a los desarrolladores a través de la publicidad.

Cabe destacar que la plataforma Firebase posee una gran cantidad de documentación y ofrece un plan gratuito muy interesante para aquellos que quieran comenzar a desarrollar sus propias aplicaciones.

Tanto la aplicación móvil como la aplicación web de este proyecto utilizan algunos de los servicios de Firebase para su correcto funcionamiento. En ambas aplicaciones, la autenticación de usuarios y la configuración de reglas



de acceso a datos son posibles gracias a Firebase Authentication [18]. En cuanto al almacenamiento de datos, se hace uso de los servicios Firebase Realtime Database [19] que permite una comunicación en tiempo real entre la aplicación móvil y la aplicación web, y Firebase Storage [20] que permite el almacenamiento de los vídeos de los usuarios de la aplicación móvil.

La idea de implementar las aplicaciones de Smart Dash Cam utilizando una plataforma actual y de constante crecimiento, capaz de ofrecer tantos servicios de calidad, convierten a Firebase en candidato ideal para el desarrollo.

### **3.3.2 Wowza Streaming Engine**

Wowza Streaming Engine [21] aparece como solución a la implementación del servidor de streaming del proyecto. En lugar de crear un servidor de streaming propio, se opta por confiar en una tecnología que es usada por más de 19000 organizaciones actuales.

Wowza Media Systems, los desarrolladores de esta tecnología, ofrecen una librería para Android que implementa diversas clases en código y facilita el desarrollo de aplicaciones, y una herramienta para la visualización de streamings. Sin embargo, con el fin de independizar el código de la aplicación móvil y la aplicación web de la tecnología servidor, se opta por utilizar una librería de terceros denominada Libstreaming para la implementación del streaming y el framework VideoJS para la visualización de la retransmisión. Esto permite desacoplar las tecnologías y da la opción para cambiar el servidor de streaming en un futuro si fuera necesario.

### **3.3.3 WampServer**

WampServer [22] se introduce en Smart Dash Cam como entorno de desarrollo web a raíz de un problema encontrado durante la realización del proyecto. Esta tecnología se compone de un conjunto de herramientas que permiten crear aplicaciones web utilizando un servidor web Apache, una base de datos MySQL y el lenguaje de programación PHP.

En un principio, la aplicación web del proyecto se basaba en una página simple desarrollada con HTML, CSS y JS, que podía ser accesible

públicamente gracias a Firebase Hosting. Este servicio de Firebase, no comentado con anterioridad, ofrece un hosting gratuito para la publicación de páginas web. Sin embargo, a pesar de que las pruebas en ámbito local fueron exitosas, en el momento de desplegar el sistema públicamente, se encontraron errores con las funcionalidades de la aplicación. Como solución rápida y sencilla a este problema surge la idea de utilizar la herramienta WampServer. De este modo, se instaló y configuró WampServer en una máquina con Windows y se desplegó la aplicación web de forma exitosa.

Al realizar el despliegue de la aplicación mediante Firebase Hosting, se proporciona un subdominio en “firebaseapp.com”. Tras descartar este servicio era necesario encontrar una forma de establecer un dominio accesible públicamente para el correcto funcionamiento de la web. Para cubrir esta necesidad el proyecto utiliza el servicio NO-IP [23].

Gracias a WampServer, NO-IP y una configuración correcta de la red, es posible acceder desde internet a la aplicación web de Smart Dash Cam.

### **3.4 Seguridad**

Los datos de los usuarios de Smart Dash Cam deben viajar por canales de comunicación seguros. La criptografía se usa para proteger la autenticidad, la integridad y la confidencialidad de la información. De este modo, se garantiza el origen de los mensajes, se asegura que estos mensajes no han sido alterados y se protege la confidencialidad. Con este fin, para implementar la seguridad en las conexiones mediante HTTPS se utiliza OpenSSL [24].

OpenSSL es un proyecto de software libre disponible para la mayoría de sistemas operativos. Ofrece un conjunto de herramientas para facilitar la implementación de protocolos relacionados con la seguridad como el Secure Sockets Layer (SSL) y el Transport Layer Security (TLS) [25], y permite crear y gestionar certificados digitales que pueden aplicarse a un servidor, como por ejemplo, el servidor Apache que incluye WampServer. Estos certificados, necesarios para trabajar con el protocolo SSL, son firmas electrónicas que demuestran la identidad al acceder a sitios web y que deben ser verificados por una autoridad certificadora (CA).

En criptografía, una autoridad certificadora se define como una entidad de confianza, responsable de emitir y revocar los certificados digitales y de verificar la identidad del solicitante de un certificado antes de su expedición. Para asegurar el cifrado de los datos en la comunicación, se han generado los certificados necesarios utilizando OpenSSL a través del siguiente proceso:

- Creación de una autoridad certificadora para tener la posibilidad de generar certificados cuando un usuario desee conectarse al servidor web.
- Creación del certificado del servidor. Se genera y se firma el certificado del servidor a través de la autoridad certificadora.
- Creación de los certificados para los clientes. Estos certificados deben ser entregados a los usuarios para ser usados desde su navegador.
- Creación de una lista de revocación, con el fin de poder revocar a través de la autoridad certificadora los certificados que hayan sido comprometidos.

### **3.5 Otras herramientas**

En el desarrollo de Smart Dash Cam se ha aplicado una metodología ágil [26]. Para ello, se ha requerido el uso de repositorios de código y herramientas de control de versiones. Todo el código de las aplicaciones se ha almacenado en la plataforma GitLab [27] y el control de versiones se ha realizado a través de Git [28]. Asimismo, se ha utilizado Trello [29] para llevar la gestión del avance del proyecto.

# Capítulo 4.

## Desarrollo

A continuación se procede a explicar con detalle el desarrollo de la aplicación móvil y la aplicación web, y las configuraciones realizadas para obtener el funcionamiento esperado del sistema Smart Dash Cam.

### 4.1 Arquitectura general del sistema

Como se ha mencionado anteriormente, este proyecto consta de una aplicación móvil y una aplicación web que utilizan los servicios de Firebase y Wowza Streaming Engine.

La aplicación móvil permite obtener datos de importancia de los usuarios, como nombre, ubicación y capturas de los accidentes, y enviarlos a los servicios de almacenamiento y base de datos en tiempo real ofrecidos por Firebase. Además, es la encargada de iniciar el streaming a través del servidor de Wowza Streaming Engine, que posteriormente podrá ser visualizado en la aplicación web.

La aplicación web permite conocer en tiempo real si un usuario requiere de asistencia gracias a la carga de datos desde Firebase. En el instante en que se detecta desde la aplicación móvil que un usuario requiere de asistencia se actualiza la interfaz de la aplicación web indicando este hecho, permitiendo la recepción del streaming desde Wowza Streaming Engine y ofreciendo la posibilidad de reproducir el vídeo del usuario accidentado y de consultar su ubicación utilizando Google Maps.

En la parte servidor, además de una máquina que ejecuta la herramienta de streaming de Wowza, existe una máquina que ejecuta WampServer y, a través de Apache, permite el despliegue de la aplicación web accesible desde un dominio establecido utilizando la plataforma NO-IP.

## 4.2 Aplicación móvil

El diseño y la implementación de la aplicación móvil constituyen la parte de mayor importancia de este proyecto. Con el objetivo de llegar al máximo número de usuarios posibles, se ha optado por desarrollar una aplicación cuidando al detalle su interfaz, sus funcionalidades y su compatibilidad con los dispositivos móviles. Para ello, se ha elegido la versión Ice Cream Sandwich (4.0.3 / API 15) del sistema operativo Android como la menor posible para el correcto funcionamiento de la aplicación, impidiendo la instalación en dispositivos con versiones anteriores. Esta limitación se debe a que la mayoría de los usuarios utilizan versiones superiores a la indicada [30] y, en versiones inferiores, se podrían encontrar problemas debido a las exigencias de algunos procesos de la aplicación, como la grabación de vídeo o el streaming.

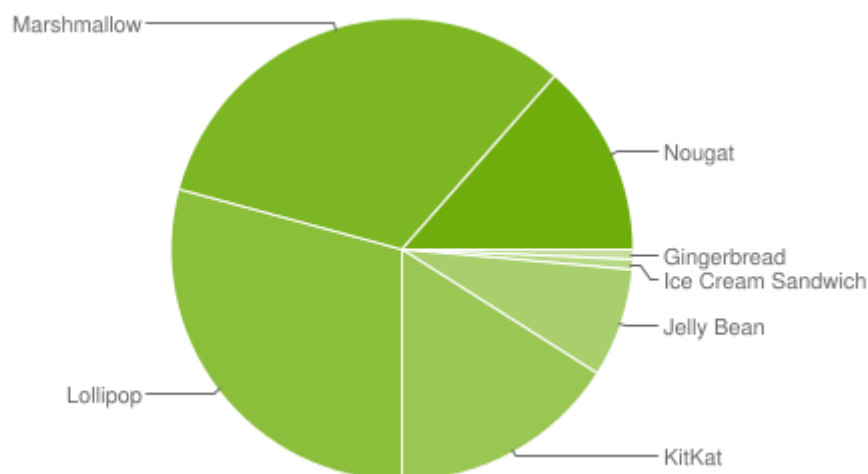


Figura 4.1. Distribución de versiones de Android.

Es de interés destacar, que utilizar para el desarrollo la versión elegida sólo afecta a los usuarios de la versión Gingerbread, siendo posible el uso de la aplicación en el resto de versiones, lo que constituye la mayor parte de usuarios de Android(ver Figura 4.1).

En lo referido a la interfaz y a pesar de que se ha mencionado que la interacción con la aplicación es mínima, no se descuida la usabilidad y se implementa un desarrollo basado en la guía de diseño de Material Design.

Aquellas pantallas donde la interacción es mayor, como son las de autenticación y configuración, cuidan más este detalle, esperando mejorar la experiencia del usuario.

En cuanto a la organización en Android Studio, la aplicación se ha estructurado en distintos paquetes y directorios con el objetivo de mejorar la limpieza y obtener un mayor rendimiento en el desarrollo. En la figura 4.2, se muestra una captura de la estructuración del proyecto.

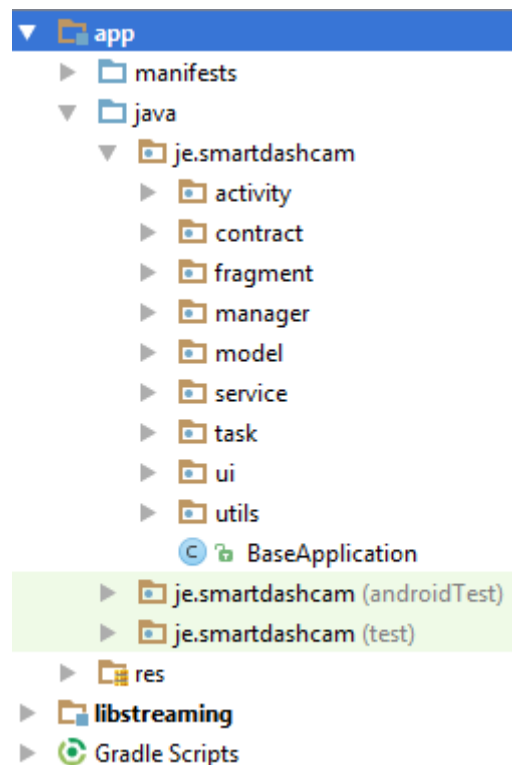


Figura 4.2. Estructura del proyecto de la aplicación móvil.

A continuación se analiza el contenido de algunos de los paquetes:

- Activity / Fragment. Contienen el código que afecta principalmente a las pantallas de la aplicación.
- Contract. Contiene las definiciones de las constantes de la aplicación.

- Manager. Contiene las clases que facilitan el acceso a funcionalidades de la aplicación, como pueden ser los sensores, la cámara, etc.
- Model. Contiene los modelos de los datos que se utilizan en ciertas funcionalidades de la aplicación, como en el almacenamiento de la información en Firebase.
- Service / Task. Contienen servicios y tareas de ejecución en segundo plano, como el servicio de Bluetooth que permite el auto-arranque.
- UI. Contiene el código de elementos de la interfaz personalizados que son creados dinámicamente.
- Utils. Contiene el código de diversas funciones de utilidad para el desarrollo de la aplicación, como aquellas que facilitan la conversión del modelo de datos en formato JSON para la transferencia a Firebase, o funciones de operaciones con ficheros y fechas.

En el directorio `res` se incluyen los documentos XML que definen las interfaces gráficas de la aplicación además de otros recursos de interés como las imágenes, los colores y las strings.

La configuración básica de la aplicación se realiza a través del documento Manifest [31]. Este documento utiliza una sintaxis XML y permite establecer ciertas características, como pueden ser, los componentes (activities, servicios o intents), los permisos o la versión de la aplicación. Para su correcto funcionamiento, Smart Dash Cam requiere de los permisos siguientes: internet, Bluetooth, localización, grabación de audio, cámara y almacenamiento.

Asimismo, la configuración de las dependencias del proyecto se realiza a través de Gradle [32]. Esta herramienta tiene como fin ayudar a la construcción, automatización y despliegue de código, facilita la instalación de las librerías necesarias para el desarrollo. A continuación, se indican las dependencias utilizadas por la aplicación móvil:

Dependencias de Google para la autenticación y la localización:

```
compile 'com.google.android.gms:play-services-auth:11.0.0'  
compile 'com.google.android.gms:play-services-location:11.0.0'
```

Dependencias de Firebase para el uso de sus servicios:

```
compile 'com.google.android.gms:play-services-auth:11.0.0'  
compile 'com.google.android.gms:play-services-location:11.0.0'
```

Dependencias de soporte de Android

```
compile 'com.android.support:design:25.2.0'  
compile 'com.android.support:appcompat-v7:25.2.0'  
compile 'com.android.support:recyclerview-v7:25.2.0'
```

Entre estas dependencias se incluyen las librerías correspondientes a los servicios de Google, los servicios de Firebase y las herramientas gráficas fundamentales para el desarrollo de la aplicación. Además, en el proyecto existen otras dependencias a librerías, que sirven de utilidad y facilitan la implementación de ciertas funcionalidades, como por ejemplo, la librería GSON [33] para la conversión de datos JSON.

En los apartados siguientes se describe el proceso que ha tenido lugar en el desarrollo de las funcionalidades principales de la aplicación móvil Smart Dash Cam.

#### 4.2.1 Autenticación

Para implementar la funcionalidad de autenticación en la aplicación móvil se hace uso de la herramienta Firebase Authentication que proporciona servicios de back-end, SDK fáciles de usar y bibliotecas de UI ya elaboradas para autenticar a los usuarios. Asimismo, admite autenticación mediante contraseñas, números de teléfono, proveedores de identidad federados populares, como Google, Facebook y Twitter, y mucho más.



Smart Dash Cam registra a los usuarios a través de los campos de correo y contraseña. Al iniciar la aplicación por primera vez se le da la posibilidad al usuario de crear una cuenta asociada y se procede al inicio de sesión con el servidor. Si la operación es exitosa, la sesión queda iniciada en el dispositivo del usuario aunque se cierre la aplicación. Para cerrar sesión, el usuario deberá utilizar la opción implementada para ese fin.

En el desarrollo de la aplicación se ha generado un fichero denominado Firebase Auth Manager que contiene las funciones necesarias para implementar la autenticación. El siguiente fragmento de código, muestra el método que permite registrar a los usuarios en la aplicación.

```
public void doSignUp(final String name, String email, String password) {
    mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    FirebaseUser user = task.getResult().getUser();
                    updateFirebaseName(name, user);
                    DatabaseUtils.addUser(user.getId(), name);
                }
            }
        });
}
```

Como se observa en el ejemplo, los parámetros email y password obtenidos en la pantalla de registro, son enviados a una de las funciones que ofrece el objeto mAuth del SDK de Firebase Authentication. A su vez, si el registro es exitoso, se realiza una llamada a la función propia de la aplicación updateFirebaseName, que permite acceder a la base de datos de Firebase y crear un registro con los datos del usuario. Este registro, servirá de nexo con la aplicación web.

#### 4.2.2 Configuración

Las opciones de configuración son bastante limitadas. Sin embargo, es necesario mencionarlas debido a que afectan a otras funcionalidades de la aplicación.

Como primera opción, se ofrece la posibilidad de establecer la duración del vídeo que graba el dispositivo móvil. Esta duración, se mide en segundos y acepta los valores de 30, 45 o 60. Limitar la duración de las grabaciones a ciertas medidas de tiempo logra evitar un consumo excesivo de la memoria del dispositivo. Además, según las pruebas realizadas, estas medidas son suficientes para entender lo ocurrido en un accidente.

Por otro lado, existe la posibilidad de vincular un dispositivo Bluetooth preferido a través de una de las pantallas de configuración. Para ello se han implementado dos funcionalidades a destacar: una funcionalidad que permite seleccionar el dispositivo a través de la lista de Bluetooth favoritos, y otra, que permite realizar una búsqueda de nuevos dispositivos. Para el tratamiento de la información, se considera necesario el desarrollo de un modelo, denominado BluetoothItem, que tiene como atributos el identificador y la dirección MAC, y que sirve para almacenar los datos proporcionados por el servicio de Bluetooth. Esta pantalla de configuración sólo será accesible si el usuario indica que desea utilizar la opción de auto-arranque mediante Bluetooth.

Tanto los datos de la configuración como otros datos que controlan el estado de la aplicación se almacenan en el dispositivo móvil. Android ofrece diferentes opciones para la persistencia de datos privados [34]. Entre ellas se distinguen: las SharedPreferences, la memoria interna del dispositivo y una base de datos SQLite. Como los datos a almacenar son del tipo clave-valor la mejor opción para implementar la persistencia de datos es utilizar las SharedPreferences. Asimismo, como el proyecto requiere del almacenamiento del vídeo del usuario, la aplicación también utiliza la memoria interna del dispositivo.

### **4.2.3 Auto-arranque**

Una de las aportaciones más llamativas de la aplicación es la funcionalidad de auto-arranque vía Bluetooth.

En el avance de la tecnología la automatización de las tareas repetitivas siempre ha estado presente. En el uso de herramientas basadas en dashcams, el usuario tiene la obligación de iniciar la aplicación cada vez que entra a su vehículo, algo que puede dejar de ser atractivo para muchos o que se puede

olvidar con facilidad. Por esta razón, surge la idea de implementar una funcionalidad que permita iniciar la aplicación automáticamente al arrancar el vehículo. Para ello, es necesario vincular el Bluetooth del vehículo previamente en la pantalla de configuración de la aplicación.

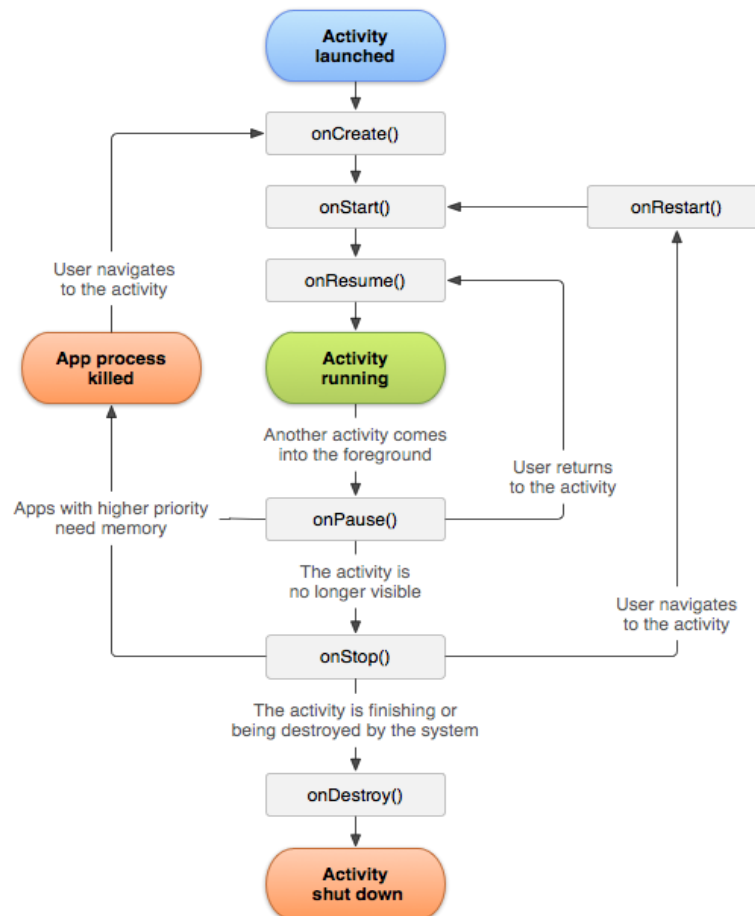


Figura 4.3. Ciclo de vida de una Activity.

En el desarrollo de esta funcionalidad surge un problema relacionado con el ciclo de vida de las aplicaciones Android (ver Figura 4.3). Una vez el usuario configura Smart Dash Cam y se inicia la pantalla de grabación de vídeo, la aplicación tiene todos los datos necesarios para su correcto funcionamiento. Estos datos se conservan mientras la aplicación esté viva, pero en el momento de su cierre, ya sea por el usuario o por el sistema operativo, los datos dejarán de persistir. En este punto, surge el siguiente problema:

1. La aplicación, iniciada previamente, ejecuta una funcionalidad para volver a iniciarse automáticamente al conectar con el Bluetooth del vehículo.
2. Tras el cierre de la aplicación se descartan todos los datos y funciones de la aplicación.
3. El inicio automático no es posible dado que el código que lanza esa funcionalidad ya no existe en memoria.

En este caso la solución encontrada es la utilización de un servicio que se ejecuta la primera vez que se inicia la aplicación y se configura con la opción *START\_STICKY*. Esta opción permite la persistencia del servicio tras el cierre de la aplicación.

El servicio implementado se mantiene a la escucha de las conexiones Bluetooth y ejecuta la orden de iniciar la aplicación en el momento que detecta el establecimiento de una conexión entre el dispositivo móvil y el Bluetooth vinculado. Para su correcto funcionamiento, el usuario debe seleccionar el Bluetooth del vehículo como favorito en la configuración de la aplicación. Las comparaciones se realizan a través de la dirección MAC única de los dispositivos.

#### **4.2.4 Rutina de grabación de vídeo**

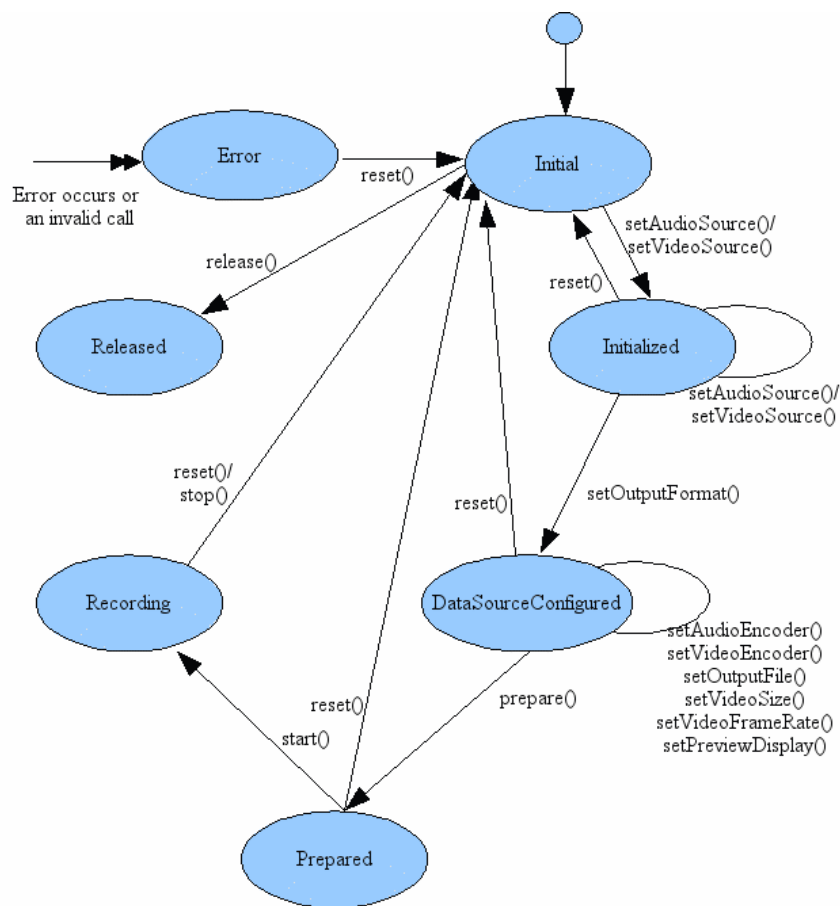
La rutina de grabación de vídeo supone una de las funcionalidades que mayor esfuerzo ha requerido en la ejecución del proyecto.

Es importante diferenciar las posibles opciones y herramientas que ofrece el SDK de Android para la realización de proyectos utilizando la cámara del dispositivo móvil.

Para obtener acceso al hardware de la cámara del dispositivo móvil se utiliza la clase `Camera`. Esta clase permite obtener una instancia de las cámaras del dispositivo y configurar una serie de parámetros como el enfoque o la orientación.

En el proceso de mostrar las imágenes obtenidas desde la cámara podemos elegir entre dos clases para el tratamiento de gráficos e imágenes denominadas `SurfaceView` y `TextureView`. Existen algunas diferencias entre ambas, como por ejemplo, el hecho de que `TextureView` permita realizar animaciones y transformaciones con las imágenes obtenidas. Sin embargo, estas funcionalidades no son necesarias para el desarrollo del proyecto y, teniendo en cuenta que `TextureView` produce un mayor consumo de recursos, se ha optado por utilizar `SurfaceView`.

Hasta este punto, haciendo uso de estas clases es posible realizar una aplicación que muestre las imágenes obtenidas a través de la cámara del dispositivo.



**MediaRecorder state diagram**

Figura 4.4. Diagrama de estados de `MediaRecorder`.

Para poder capturar estas imágenes es necesario utilizar la clase `MediaRecorder` que ofrece funciones para la implementación de procesos de grabación de audio y vídeo en Android. La documentación oficial, indica que el objeto creado con `MediaRecorder` debe seguir una configuración específica, y pasar por una serie de estados antes de comenzar la ejecución de la grabación (ver Figura 4.4). Es importante tener esto en cuenta, ya que cualquier intento de iniciar una grabación con un objeto que no ha sido inicializado correctamente, causará un error en la aplicación.

En la implementación del bucle de grabación de vídeo es necesario utilizar un listener del objeto `MediaRecorder` llamado `OnInfoListener`. De este modo, es posible recibir información acerca de los cambios del estado en el objeto. A continuación se muestra un fragmento de código de la implementación.

```
public void initMediaRecorder() {
    mMediaRecorder = new MediaRecorder();
    mMediaRecorder.setOnInfoListener(new MediaRecorder.OnInfoListener() {
        @Override
        public void onInfo(MediaRecorder mr, int what, int extra) {
            switch (what) {
                case MediaRecorder.MEDIA_RECORDER_INFO_MAX_DURATION_REACHED:
                    isRecording = false;
                    startRecorder();
            }
        }
    });
}
```

Al iniciar la pantalla de grabación de vídeo, el objeto `MediaRecorder` se inicializa con la duración establecida por el usuario en la pantalla de configuración y comienza a realizar una primera grabación. Cuando alcanza el máximo de esta duración se vuelve a lanzar el proceso, lo que permite obtener el funcionamiento deseado.

Además, `MediaRecorder` incluye otros parámetros de configuración como la calidad del vídeo, la duración máxima, y el fichero de salida que contiene la grabación realizada. En el dispositivo móvil la aplicación crea un directorio para almacenar este fichero y cada vez que se inicia una nueva grabación se sobrescribe para evitar un consumo excesivo de memoria.

#### **4.2.5 Detección de colisiones**

A través del acelerómetro del dispositivo móvil es posible realizar un algoritmo capaz de conocer si un usuario ha sufrido un accidente. Estos sensores están pensados para medir la aceleración proporcionando una señal eléctrica según la variación física detectada. De este modo, se implementa un algoritmo para detectar un cambio brusco en la aceleración que supere 2.7G, considerando como colisión cualquier fuerza superior este valor.

En este desarrollo ha sido necesario crear un filtro para aislar la fuerza de la gravedad y poder obtener medidas reales de la aceleración del dispositivo. En el fichero ShakeManager de la aplicación Android, se puede consultar la implementación del algoritmo utilizado.

Una vez se detecta el accidente se procede a la interacción con el usuario.

#### **4.2.6 Interacción por voz**

En el momento de producirse un accidente, la aplicación muestra un diálogo y consulta al usuario si requiere de asistencia. Llegado este punto, el usuario tiene la posibilidad de realizar una respuesta mediante touch o una respuesta mediante voz.

La funcionalidad de interacción por voz pretende facilitar la interacción del usuario con la aplicación durante la conducción, siendo de utilidad en el momento de solicitar asistencia. Para su implementación, se utilizan las herramientas que ofrece Android en reconocimiento de voz y conversión de texto a audio. Estas herramientas son accesibles a través de las clases SpeechRecognizer y TextToSpeech.

Por medidas de seguridad, si pasados unos segundos no se obtiene la respuesta del usuario, se considera que este requiere de asistencia y se procede al envío de datos y al inicio del streaming.

#### **4.2.7 Envío de datos**

La aplicación móvil Smart Dash Cam genera y almacena datos del usuario. En caso de accidente, estos datos se envían a Firebase, desde donde son accesibles para la plataforma web.

Con el fin de facilitar el proceso, se crea un modelo de datos en la aplicación móvil con los datos del usuario, y se convierten a formato JSON utilizando la librería GSON. Esta librería, ofrece funciones para la conversión entre objetos Java y JSON.

El modelo de datos incluye el nombre del usuario, su última localización y una variable booleana que indica si el usuario ha sufrido un accidente. Previamente al almacenamiento de estos datos en el servicio Firebase Realtime Database, se agrega la ubicación del usuario obtenida a través de las herramientas de localización de Android.

El envío de la localización del usuario, es un requisito indispensable de la aplicación Smart Dash Cam. Para la implementación de esta funcionalidad, se utiliza FusedLocationProvider, una herramienta que trabaja con los servicios de localización de Google.

```
mFusedLocationClient.getLastLocation()
    .addOnSuccessListener(this, new OnSuccessListener<Location>()
{
    @Override
    public void onSuccess(Location location) {
        if(location != null){
            BaseApplication.getUser().setLastLocation(location);
        }
    }
});
```

Como se puede observar en el fragmento de código, en el momento de detectar un cambio en la ubicación del usuario, se modifica el modelo de datos. De forma instantánea, la aplicación actualiza ese valor en Firebase, otorgando fiabilidad a los datos mostrados en la aplicación web.

Para llevar a cabo la actualización de los datos en Firebase Realtime Database se utilizan las clases DatabaseReference y FirebaseDatabase incluidas en el SDK de Firebase. Estas clases ofrecen funciones que permiten obtener referencias a nodos de la base de datos y actualizar sus valores, logrando la sincronización en tiempo real de todas las aplicaciones implicadas. A continuación se muestra un ejemplo de acceso al servicio Firebase Realtime Database.



```

public static void updateUser() {
    DatabaseReference databaseReference = FirebaseDatabase
        .getInstance()
        .getReference();

    databaseReference.child("users")
        .child(BaseApplication.getUid())
        .setValue(BaseApplication.getUser());
}

```

Además de los datos del usuario, la aplicación genera una grabación de vídeo, que almacena en un directorio del dispositivo móvil. Cuando se procede a realizar el envío de datos, se sube una copia a Firebase Storage haciendo uso de las clases `FirebaseStorage` y `StorageReference`. Estas clases, permiten obtener una referencia al servicio de almacenamiento y enviar los vídeos especificando la ruta deseada. Además, es posible asignar información adicional al archivo enviado a Firebase Storage en forma de metadatos [35].

```

StorageMetadata metadata = new StorageMetadata.Builder()
    .setContentType(METADATA_CONTENT_TYPE)
    .setCustomMetadata(METADATA_LOCATION, location)
    .setCustomMetadata(METADATA_TIME, timeStamp)
    .build();

```

Por último, es importante destacar, que las búsquedas de usuarios y de sus directorios, se realizan en la aplicación por medio del id de autenticación. En el proceso de registro de usuarios, se lanza un procedimiento que crea una entrada con el id del usuario en la base de datos. Este campo actúa como nodo para almacenar el resto de los datos del usuario. Mientras el usuario permanezca autenticado, la aplicación utiliza este id como clave en sus accesos a los servicios de Firebase.

#### 4.2.8 Streaming

La funcionalidad de streaming es otro de los puntos fuertes del proyecto Smart Dash Cam, ya que permite conocer la situación de los afectados en tiempo real.

La implementación de esta característica, se realiza utilizando una librería de terceros denominada Libstreaming. Esta librería se adapta perfectamente a los requisitos y tecnologías del proyecto y permite realizar streaming desde los dispositivos Android utilizando el protocolo RTP [36]. Además, funciona para versiones superiores a la 4.0 y soporta codificación en los formatos H.264 [37], H.263 [38], AAC [39] y AMR [40].

Para desarrollar la funcionalidad de streaming, es necesaria la utilización de dos clases: la sesión y el cliente. El objeto de sesión contiene la configuración de los parámetros para la sesión de streaming, como por ejemplo, el formato de codificación de vídeo. El objeto cliente permite establecer el destino de la retransmisión, siendo en este caso, el servidor Wowza Streaming Engine.

Este código muestra los valores de configuración para la sesión:

```
mSession = SessionBuilder.getInstance()
    .setContext(getActivity().getApplicationContext())
    .setAudioEncoder(SessionBuilder.AUDIO_AAC)
    .setAudioQuality(new AudioQuality(8000, 16000))
    .setVideoEncoder(SessionBuilder.VIDEO_H264)
    .setSurfaceView(mSurfaceView)
    .setCallback(mSessionCallback).build();
```

Este código muestra los valores de configuración para el cliente:

```
mClient.setCredentials(Contract.PUBLISHER_USERNAME,
                        Contract.PUBLISHER_PASSWORD);
mClient.setServerAddress(ip, Integer.parseInt(port));
mClient.setStreamPath("/" + path);
```

En cuanto a la configuración de la sesión se ha elegido el formato H.264 para la codificación del vídeo debido a la calidad de compresión y esperando obtener el mejor rendimiento posible. Asimismo, en la configuración del cliente, las credenciales y la ip especificadas corresponden con los parámetros del servidor de Wowza Streaming Engine, mientras que el path corresponde al identificador del usuario.

## 4.3 Aplicación web

A pesar de que la aplicación móvil es el desarrollo más destacado de este proyecto, no sería útil sin la existencia de una aplicación web que permitiera a los servicios de emergencia consultar los datos de los usuarios.

Como se ha comentado con anterioridad, la aplicación web desarrollada consiste en una aplicación sencilla utilizando HTML, CSS y JS.

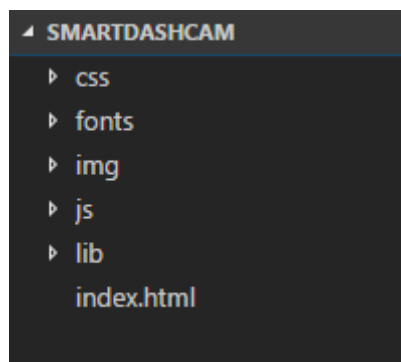


Figura 4.5. Estructura del proyecto de la aplicación web

En cuanto a la organización en Visual Studio Code (ver Figura 4.5), la aplicación se ha estructurado de forma similar a otros proyectos de desarrollo web. A continuación, se explica el contenido de los directorios más importantes:

- CSS. Este directorio contiene los códigos de estilo de la aplicación web.
- JS. Este directorio contiene los archivos JavaScript que dotan a la aplicación de funcionalidad.
- LIB. Este directorio contiene las librerías utilizadas en el proyecto.

En los directorios CSS y LIB se almacenan los archivos de Materialize, una librería que permite el desarrollo de interfaces basadas en Material Design. Asimismo, en esta estructura de la aplicación, se incluye la única página HTML del proyecto, que contiene la definición de los elementos y la carga de

otras funcionalidades de importancia, como la librería que permite utilizar el reproductor VideoJS y las necesarias para acceder a los datos de Firebase.

Este código muestra la carga de las librerías de VideoJS:

```
<script src="http://vjs.zencdn.net/5.19.2/video.js"></script>
```

Este código muestra la carga de librerías y la configuración de Firebase:

```
<script src="https://www.gstatic.com/firebasejs/4.1.3/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyCCjN92EdEyXKqtYqVj91ezbihFlmygQeU",
    authDomain: "smartdashcam.firebaseio.com",
    databaseURL: "https://smartdashcam.firebaseio.com",
    projectId: "smartdashcam",
    storageBucket: "smartdashcam.appspot.com",
    messagingSenderId: "45139350227"
  };
  firebase.initializeApp(config);
</script>
```

Destacar que la configuración para el uso de los servicios de Firebase se obtiene añadiendo la aplicación a un nuevo proyecto en la consola de su plataforma web. A continuación, se describe el desarrollo de las funcionalidades que ofrece la aplicación web.

### 4.3.1 Carga y actualización de datos

La carga de los datos de los usuarios de la aplicación móvil es posible gracias al uso de los servicios de Firebase Realtime Database y Firebase Storage.

Para acceder a la base de datos y al almacenamiento de Firebase desde la aplicación web se utilizan los objetos `firebase.database()` y `firebase.storage()`,

respectivamente. Tal y como ocurre en la aplicación móvil, estas clases ofrecen las funciones necesarias para acceder a los datos de los usuarios.

Este código muestra un acceso al servicio Firebase Realtime Database:

```
firebase.database().ref("users/").on("child_changed", function (data) {
    updateUserListView(data);
});
```

Haciendo uso de los listeners que ofrecen las clases anteriores, es posible detectar cambios en los datos de los usuarios. Esto supone un gran impacto en la experiencia del usuario y en el funcionamiento del sistema Smart Dash Cam, siendo posible recibir actualizaciones en tiempo real desde los dispositivos móviles y modificar al instante la interfaz de la aplicación web para ofrecer a los servicios de emergencias una solución óptima. En el ejemplo anterior, se muestra una actualización de la interfaz de la aplicación web mediante la implementación del listener on(“child\_changed”).

### 4.3.2 Streaming

La aplicación web es encargada de ofrecer a los servicios de emergencias, una interfaz capaz de mostrar, en tiempo real, la situación de las víctimas de accidentes de tráfico. Con este objetivo, se procede a la implementación de la funcionalidad de visualización del streaming, que debe ser iniciado previamente, por los usuarios de la aplicación móvil.

Para lograr visualizar el streaming a través de la aplicación web es necesario utilizar una tecnología de reproducción de vídeo. Tras la búsqueda de información y analizando las posibilidades actuales, se elige VideoJS para este fin [41].

Como se ha mencionado con anterioridad, para acceder a las características de VideoJS y comenzar con la implementación, es necesario disponer de los ficheros JavaScript que incluyen las funciones específicas para hacer uso de esta tecnología. A partir de este punto, es posible desarrollar una función que permita capturar el streaming desde el servidor de Wowza Streaming Engine

y visualizarlo a través de la página web. Para ello, se define en el documento HTML la etiqueta `<video></video>` y se configura e inicia el reproductor en el documento JavaScript indicando el origen de la transmisión y el formato, tal y como se indica a continuación.

```
var source = "rtmp://79.152.87.184:1935/SmartDashCam/" + userId;
player.src({ type: 'rtmp/mp4', src: source });
player.play();
```

Como se puede observar, la dirección de origen depende del id de autenticación del usuario en Firebase, lo que permite obtener más de una retransmisión de forma simultánea.

### 4.3.3 Vídeo del accidente y ubicación del usuario

Además de la visualización de streaming, la aplicación web permite reproducir el vídeo del accidente y obtener la ubicación del usuario.

El servicio Firebase Storage proporciona una utilidad que permite generar enlaces para acceder al contenido de los archivos almacenados. De este modo es posible implementar una funcionalidad en la aplicación web para reproducir los vídeos de los usuarios afectados.

Este código muestra el acceso al servicio de Firebase Storage:

```
firebase.storage().ref().child(userId + "/CRASH.mp4")
  .getDownloadURL().then(function (url) {
    $(page).find(".user-streaming-video").attr("href", url);
    $(page).find(".user-streaming-video").css("visibility", "visible");
  });
```

Como se puede observar en el código anterior, el acceso al servicio de almacenamiento se realiza a través del id del usuario. Tras obtener la

referencia al vídeo se habilita un enlace en la página web para su visualización.

La ubicación del usuario se obtiene a través de un acceso al servicio Firebase Realtime Database. Con los datos de localización es posible generar un enlace a Google Maps de forma sencilla.

```
firebase.database().ref('/users/' + userId + "/lastLocation")
    .once('value').then(function (location) {
        $(page).find(".user-streaming-location").css("visibility", "visible");
        if (location !== null) {
            $(page).find(".user-streaming-location")
                .attr("href", googleMapsUrl(location));
        }
    });
```

## 4.4 Servidor

Como se ha mencionado con anterioridad, para el desarrollo de este proyecto es necesario el uso de tecnologías que permitan comunicar los datos entre la aplicación móvil y la aplicación web. Este apartado describe las configuraciones implementadas en las tecnologías de servidor con más relevancia para el proyecto: Firebase y Wowza Streaming Engine.

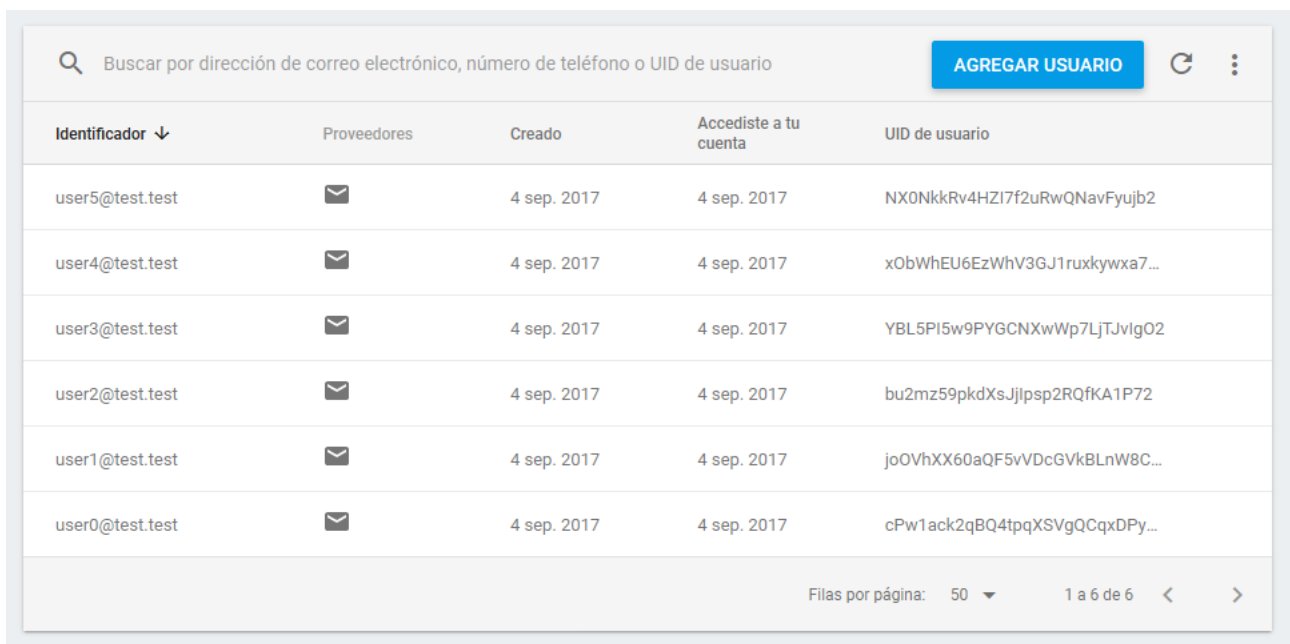
### 4.4.1 Configuración de Firebase

El primer paso en la configuración de Firebase es la creación de una cuenta de Google para tener acceso a la plataforma. Una vez realizado este paso, se crea un nuevo proyecto y se agregan las aplicaciones implicadas, en este caso, la aplicación móvil y la aplicación web.

A continuación, se añaden las dependencias y la configuración que permiten el acceso por parte de las aplicaciones a los servicios de Firebase. Tanto las dependencias como las credenciales de configuración de acceso se obtienen al añadir las aplicaciones a un proyecto en la plataforma.

Desde este momento, es posible utilizar en las aplicaciones los servicios de Firebase que se requiera. Para el desarrollo de este proyecto se utiliza Firebase Authentication, Firebase Realtime Database y Firebase Storage:

Firebase Authentication es el servicio que permite la autenticación en la aplicación móvil. Aunque ofrece diversas funcionalidades, no ha requerido configuración por parte del servidor.



The screenshot shows the Firebase Authentication console interface. At the top, there is a search bar with the text "Buscar por dirección de correo electrónico, número de teléfono o UID de usuario" and a blue button labeled "AGREGAR USUARIO". Below the search bar is a table with the following columns: "Identificador", "Proveedores", "Creado", "Accediste a tu cuenta", and "UID de usuario". The table contains six rows of user data, all created on "4 sep. 2017". At the bottom right of the table, there is a pagination control showing "Filas por página: 50" and "1 a 6 de 6".

Identificador ↓	Proveedores	Creado	Accediste a tu cuenta	UID de usuario
user5@test.test	✉	4 sep. 2017	4 sep. 2017	NX0NkkRv4HZI7f2uRwQNavFyujb2
user4@test.test	✉	4 sep. 2017	4 sep. 2017	xObWhEU6EzWhv3GJ1ruxkywxa7...
user3@test.test	✉	4 sep. 2017	4 sep. 2017	YBL5PI5w9PYGCNXwWp7LJTJvlgO2
user2@test.test	✉	4 sep. 2017	4 sep. 2017	bu2mz59pkdXsJjipsp2RQfKA1P72
user1@test.test	✉	4 sep. 2017	4 sep. 2017	joOVhXX60aQF5vVDcGVkBLnW8C...
user0@test.test	✉	4 sep. 2017	4 sep. 2017	cPw1ack2qBQ4tpqXSVgQCqxDPy...

Figura 4.6. Firebase Authentication.

Firebase Realtime Database es el servicio que usa el proyecto como base de datos en tiempo real y permite la sincronización instantánea entre la aplicación móvil y la aplicación web. Los datos se almacenan en formato JSON y siguen el esquema que se muestra en la figura 4.7.



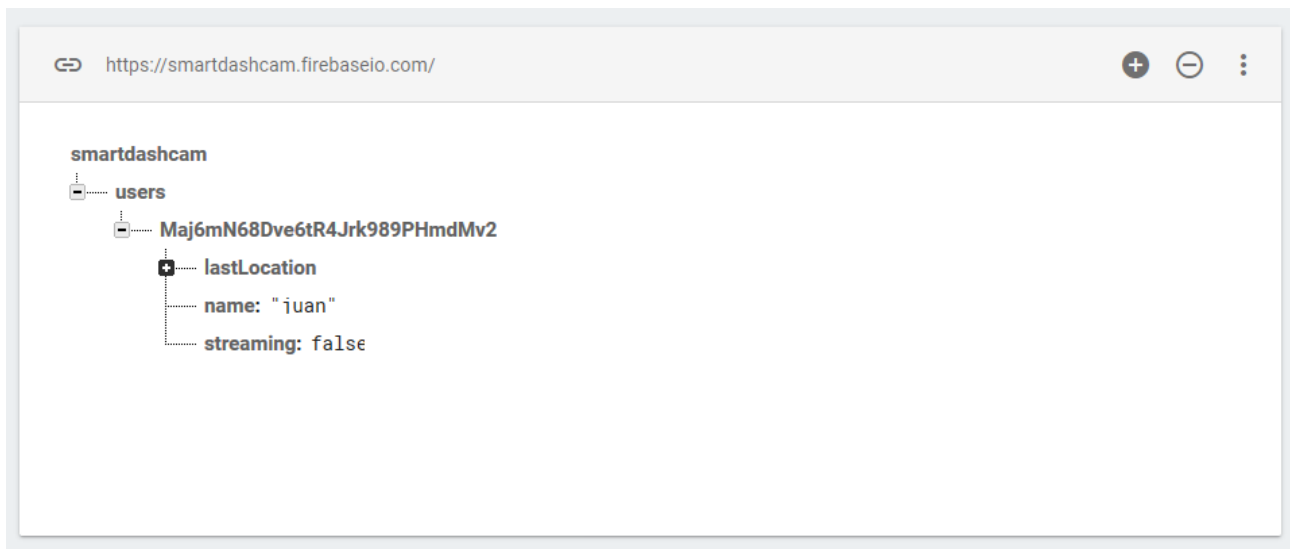


Figura 4.7. Firebase Realtime Database

Firestore se utiliza en el proyecto para almacenar los vídeos de los usuarios enviados desde la aplicación móvil.

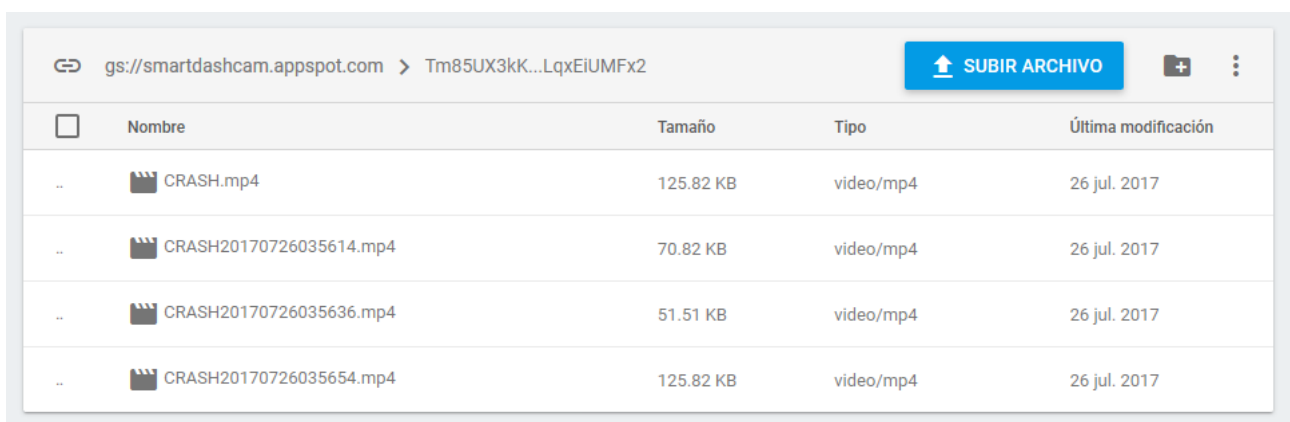


Figura 4.8. Firebase Storage.

Como se puede observar, tanto en Firebase Realtime Database como en Firestore, se utiliza el id de la autenticación del usuario para identificarlo de forma única.

La aplicación móvil es la encargada de actualizar, de forma automatizada, los datos en Firestore, mientras que la aplicación web sólo realiza accesos para consultar la información.

## **4.4.2 Configuración de Wowza Streaming Engine**

El primer paso para configurar Wowza Streaming Engine como servidor de streaming es descargar e instalar la aplicación de escritorio. Como la herramienta requiere de una licencia para su funcionamiento, se ha creado una cuenta de prueba en la plataforma para su utilización en este proyecto.

Wowza Media Systems proporciona una interfaz de configuración del servidor de streaming llamada Wowza Streaming Engine Manager. La configuración inicial es sencilla y se basa en crear las credenciales para el acceso y añadir las aplicaciones que harán uso del servicio. Estas credenciales son las que utiliza Libstreaming en la aplicación móvil para realizar el streaming con el servidor.

La herramienta proporciona múltiples opciones de configuración como el número de conexiones simultáneas o los formatos permitidos. Sin embargo, se ha optado por una configuración por defecto para la realización del proyecto.

## **4.5 Seguridad**

Este apartado describe las medidas de seguridad implementadas en el desarrollo del proyecto.

### **4.5.1 Seguridad en aplicaciones**

La seguridad en las aplicaciones se limita a la existencia de la autenticación de usuarios mediante correo y contraseña, y el cifrado de las comunicaciones a través de Firebase.

### **4.5.2 Seguridad en Firebase**

Firebase integra en sus aplicaciones una capa de seguridad en las comunicaciones utilizando el protocolo SSL, y ofrece la posibilidad de crear medidas de seguridad para la gestión de los usuarios y el acceso a los datos. Entre estas medidas de seguridad, destacan las reglas aplicables a Firebase Realtime Database y Firebase Storage que se han implementado en este proyecto. En las figuras 4.9 y 4.10 se muestran las reglas de seguridad para el acceso a los datos en estos servicios.

```
1 {
2   "rules": {
3     ".read": true,
4     ".write": "auth != null"
5   }
6 }
7
```

Figura 4.9. Reglas de seguridad en Firebase Realtime Database.

```
1 service firebase.storage {
2   match /b/{bucket}/o {
3     match /{userId}/{allPaths=**} {
4       allow read;
5       allow write: if request.auth.uid == userId;
6     }
7   }
8 }
```

Figura 4.10. Reglas de seguridad en Firebase Storage.

De este modo se consiguen aplicar reglas de seguridad para la lectura y escritura de los datos. El permiso de lectura, se le otorga a toda las aplicaciones que utilicen la dirección del servidor. Por otra parte, el permiso de escritura varía dependiendo del servicio que se utilice. En el caso de Firebase Realtime Database, se concede cuando se está autenticado, mientras que en Firebase Storage, se concede sólo cuando se es el propietario del directorio. No se aplica un mayor nivel de seguridad para el permiso de lectura, debido a que la aplicación web no dispone de autenticación, siendo necesaria la configuración actual para un correcto acceso a los datos.

# Capítulo 5.

## Presupuesto

En este capítulo se realiza un presupuesto general del coste que supone el desarrollo de Smart Dash Cam en su totalidad.

### 5.1 Personal

La siguiente tabla muestra la información sobre el presupuesto establecido para el desarrollo del proyecto, en relación al coste del personal. Para el cálculo de los costes se toma como referencia el sueldo promedio de un Ingeniero Informático en España, siendo este de un total de 14 € por hora.

Actividad	Horas	Coste
Investigación	60	840,00 €
<b>Desarrollo Móvil</b>		
Interfaz	40	560,00 €
Registro y Autenticación	20	280,00 €
Grabación de vídeo	30	420,00 €
Detección de accidentes	15	210,00 €
Interacción por voz	10	140,00 €
Operaciones con datos	20	280,00 €
Streaming	25	350,00 €
Auto-arranque	15	210,00 €
<b>Desarrollo Web</b>		
Interfaz	20	280,00 €
Operaciones con datos	15	210,00 €
Streaming	20	280,00 €
Mostrar Vídeo y Ubicación	15	210,00 €
<b>Configuración</b>		
Firebase	30	420,00 €
Wowza Streaming Engine	20	280,00 €
WampServer	15	210,00 €
NO-IP	10	140,00 €
<b>Pruebas</b>		

Aplicación Móvil	80	1.120,00 €
Aplicación Web	40	560,00 €
<b>TOTAL</b>		<b>7.000,00 €</b>

Tabla 5.1. Presupuesto del personal.

## 5.2 Componentes

La siguiente tabla desglosa los costes de los dispositivos y las licencias que intervienen en el desarrollo y funcionamiento del proyecto.

Componentes	Unidades	Coste
IDE Android Studio	1	0,00 €
Licencia de desarrollador de Android	1	22,00 €
Dispositivo Móvil	1	150,00 €
Soporte Vehículo	1	10,00 €
<b>TOTAL</b>		<b>182,00 €</b>

Tabla 5.2. Presupuesto de los componentes.

## 5.3 Coste total

La siguiente tabla muestra el coste total del proyecto:

Personal	7.000,00 €
Componentes	182,00 €
<b>TOTAL</b>	<b>7.182,00 €</b>

Tabla 5.3. Presupuesto total.

Según estos cálculos, el coste del proyecto es de un total de 7182 €. Es necesario destacar que en este presupuesto no se ha incluido el coste del servidor de streaming Wowza Streaming Engine. La plataforma dispone de una versión de prueba, pero si se quisiera llevar a un entorno real, sería necesario adquirir la versión de pago, cuyo precio es de 1995 € al año por licencia, o en otro caso, buscar alternativas que cubran esa funcionalidad.

# Capítulo 6.

## Conclusiones y líneas futuras

Este proyecto ha sido posible gracias al esfuerzo y la pasión por el desarrollo móvil.

Como resultado del trabajo realizado, se ha completado el desarrollo de un sistema formado por una aplicación móvil y una aplicación web, que aporta múltiples funcionalidades para servir de apoyo a los servicios de emergencia.

Se ha implementado una aplicación móvil con una interfaz basada en Material Design que recoge las funcionalidades de autenticación, auto-arranque vía Bluetooth, grabación de vídeo, detección de colisiones, cálculo de la ubicación del usuario, interacción por voz y streaming. La aplicación se ha diseñado cuidando al detalle la comunicación entre los servicios implicados para conseguir el mayor rendimiento y estabilidad posibles.

Se ha implementado una aplicación web que permite obtener los datos de los usuarios de la aplicación móvil y visualizar la emisión del streaming y del vídeo anterior al accidente, y que soporta actualizaciones de la información en tiempo real.

Se han implementado soluciones de servidor utilizando tecnologías actuales y de calidad como Firebase para los servicios de autenticación, base de datos en tiempo real y almacenamiento de vídeos; y Wowza Streaming Engine para el servicio de streaming.

Aun así, como en todo desarrollo, siempre pueden implementarse mejoras o nuevas funcionalidades. A continuación, se presentan una serie de ideas de trabajos futuros que pueden mejorar el sistema propuesto:

- Opciones de configuración en la aplicación móvil. La aplicación móvil no dispone de muchas opciones de configuración. Se pueden añadir nuevas características como la selección de la calidad de la grabación de vídeo y otras opciones que mejoren la usabilidad.

- Servidor de streaming. Wowza Streaming Engine no es un servicio gratuito. Se pueden investigar otras soluciones más económicas o implementar un servidor de streaming propio.
- Mejorar la seguridad. Se pueden utilizar protocolos que apliquen medidas de seguridad en la transmisión del streaming.
- Sistema de calibración para el acelerómetro. Actualmente la aplicación móvil implementa un algoritmo genérico para la detección de accidentes. Sin embargo, no todos los dispositivos utilizan el mismo hardware y las mediciones del acelerómetro pueden variar de un dispositivo a otro. Se puede añadir una funcionalidad de calibración que mejore la precisión.
- Más opciones en la aplicación web. Smart Dash Cam permite consultar la situación de los usuarios desde la aplicación web pero no permite interactuar con ellos. Se puede implementar una funcionalidad para iniciar una comunicación directa con aquellos usuarios que necesiten asistencia.

Smart Dash Cam puede llegar a tener un gran impacto en la seguridad vial y se espera que algún día pueda desarrollarse en un entorno real.

# Capítulo 7.

## Conclusions and future Work

The constant efforts and the passion for the mobile development have allowed to make this project.

As a result of the work done, the development of a system consisting on a mobile application and a web application, which provides multiple functionalities to assist the emergency services, has been completed.

The mobile application uses a Material Design based interface and contains the features of authentication, auto-boot using Bluetooth, video recording, collision detection, user location, voice interaction and streaming. The communication between the services of this application has been carefully designed to achieve the best performance and stability.

The web application allows emergency services to obtain the user data and streaming from the mobile application, supporting real-time updates.

Server solutions use technologies such as Firebase for authentication services, real-time database and video storage; and Wowza Streaming Engine for streaming service. These technologies help to provide a better system performance.

Even so, as in others developments, improvements or new features can be implemented. The following features are examples of future work that can improve the system:

- Settings in the mobile application. The mobile application have just a few configuration options. New features such as selection of video recording quality and other usability options could be added.
- Streaming Server. Wowza Streaming Engine is not free. Finding cheaper solutions or implementing a streaming server might be better.



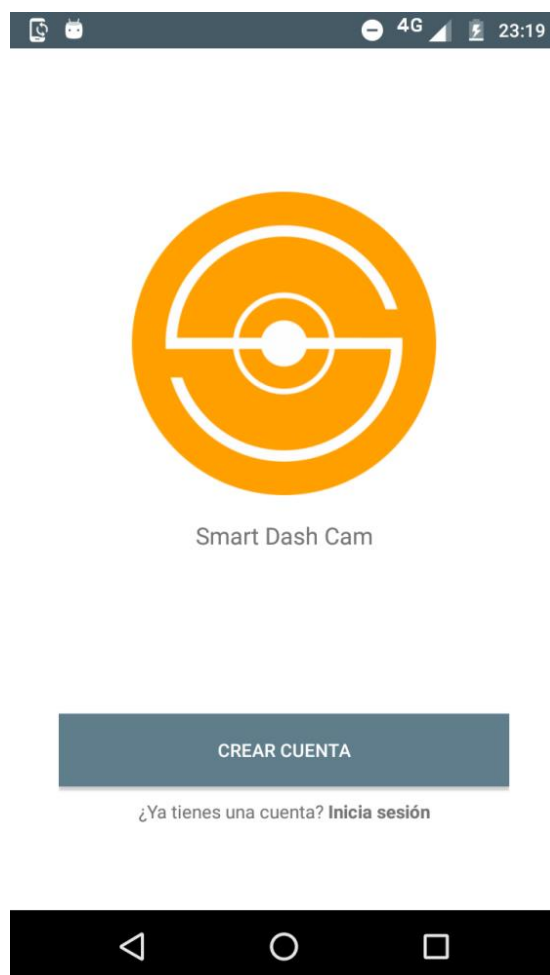
- Security improvements. New protocols for the streaming service could be implemented.
- Accelerometer calibration system. The mobile application implements a generic algorithm for traffic collision detection. However, different devices use different hardware and accelerometer measurements may differ between devices. A new calibration functionality that improves accuracy could be implemented.
- More options for the web application. Smart Dash Cam allows to check the user status from the web application but not allow to interact with them. A functionality that establishes a direct communication with the users who need assistance could be implemented.

Smart Dash Cam can have a major impact on road traffic safety and I hope that someday it would be developed in a real environment.

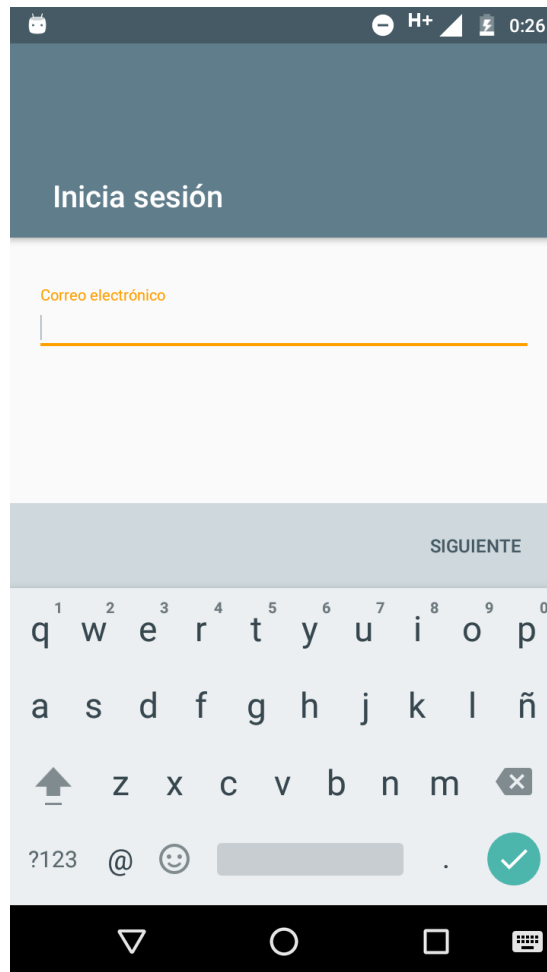
# Apéndice A.

## Interfaz de la aplicación móvil

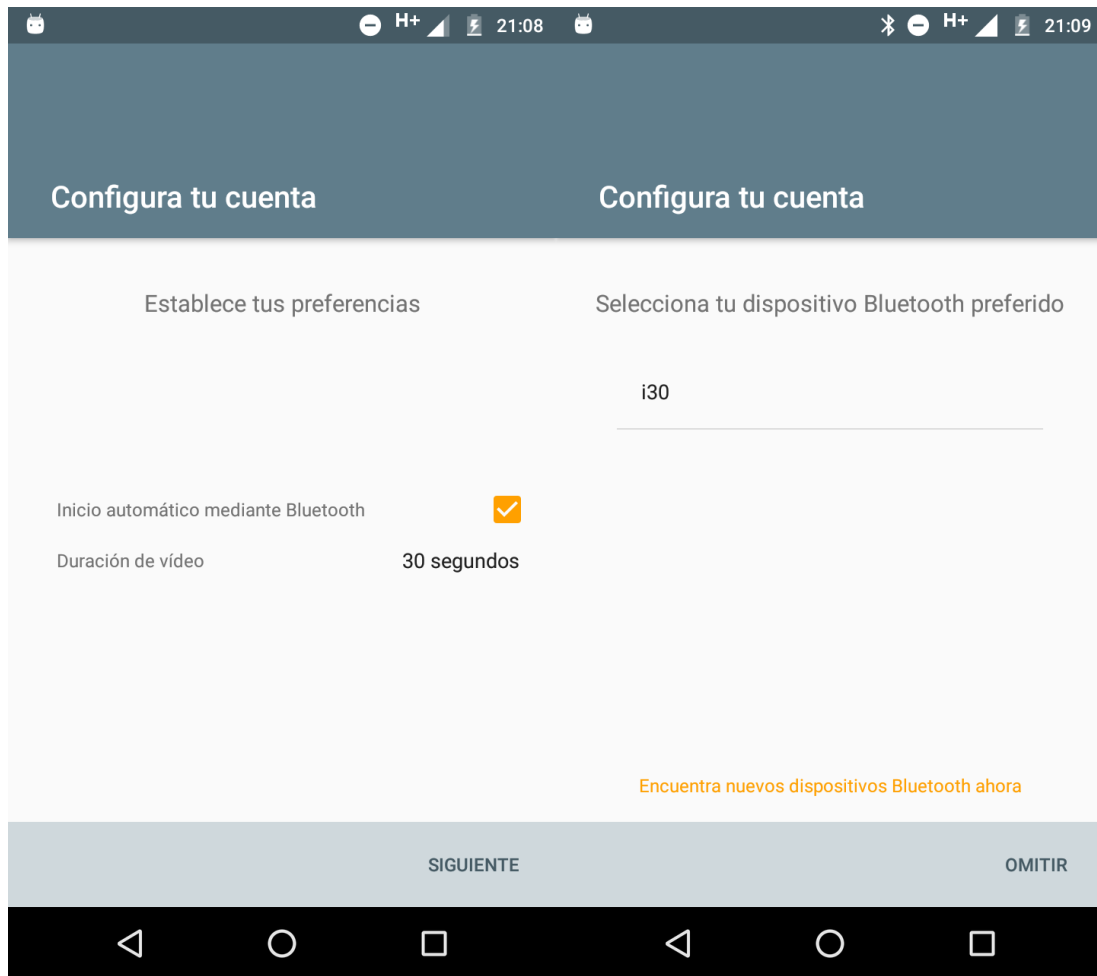
### A.1. Pantalla de autenticación / registro



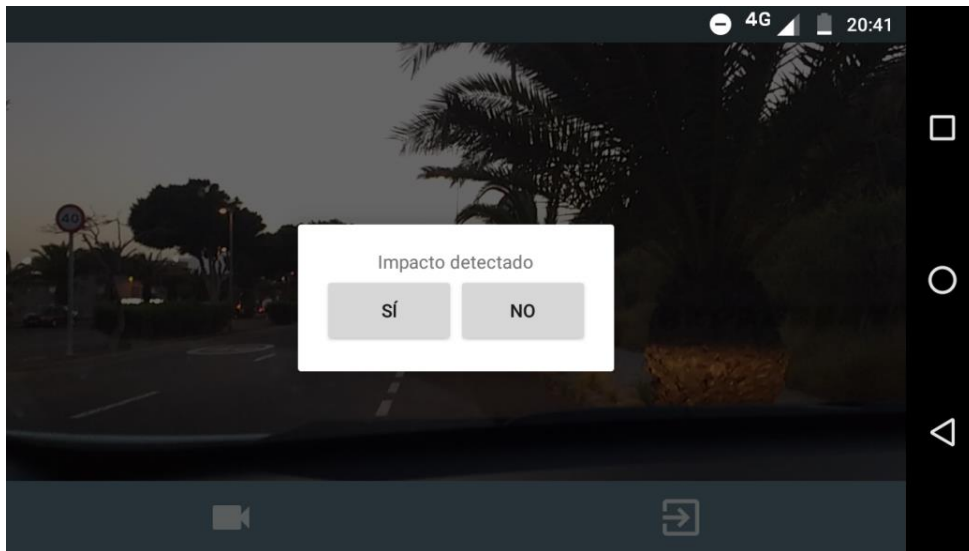
## A.2. Pantalla de formulario



## A.3. Pantallas de configuración



## A.4. Pantalla principal (grabación de vídeo)



# Apéndice B.

## Interfaz de la aplicación web

### B.1. Vista de la aplicación web



# Bibliografía

- [1] Anuario estadístico de accidentes. DGT. <http://www.dgt.es/es/seguridad-vial/estadisticas-e-indicadores/publicaciones/anuario-estadistico-accidentes/>
- [2] ¿Qué es una DashCam y para qué sirve? Álvaro Cervilla, Gizlogic. <https://www.gizlogic.com/que-es-una-dashcam-y-para-que-sirve/>
- [3] Balance de accidentes en 2016. DGT. <http://revista.dgt.es/es/noticias/nacional/2017/01ENERO/0103balance-accidentes-2016.shtml>
- [4] Google Play. <https://play.google.com/store/search?q=dashcams&c=apps&hl=es>
- [5] Google Talkback. <https://www.androidcentral.com/what-google-talk-back>
- [6] AEPD. <https://www.agpd.es/portalwebAGPD/index-ides-idphp.php>
- [7] ¿Son legales las Dash Cam? Javierin, Javierin. <https://javierin.com/legales-las-dash-cam/>
- [8] Android ya es el Sistema operativo más usado del mundo. José Mendiola Zuriarrain, EL PAÍS. [https://elpais.com/tecnologia/2017/04/04/actualidad/1491296467\\_396232.html](https://elpais.com/tecnologia/2017/04/04/actualidad/1491296467_396232.html)
- [9] Cómo instalar el Android SDK y para qué nos sirve. Enrique Pérez, Xataka Android. <https://www.xatakandroid.com/programacion-android/como-instalar-el-android-sdk-y-para-que-nos-sirve>
- [10] Material Design. <https://material.io/guidelines/>
- [11] Google Play Services: What is It and What is It for?. Chris Marshall, Android Pit. <https://www.androidpit.com/google-play-services-what-is-it-and-what-is-it-for>
- [12] Libstreaming. <https://github.com/fyhertz/libstreaming>
- [13] MEAN Framework. <http://mean.io/>
- [14] jQuery. <https://jquery.com/>

- [15] Materialize. <http://materializecss.com/>
- [16] VideoJS, The Player Framework. <http://videojs.com/>
- [17] ¿Qué es Firebase? La mejorada plataforma de desarrollo de Google. Jose Angel Zamora, El Androide libre.  
<https://elandroidelibre.elespanol.com/2016/05/firebase-plataforma-desarrollo-android-ios-web.html>
- [18] Firebase Authentication. <https://firebase.google.com/docs/auth/?hl=es-419>
- [19] Firebase Realtime Database.  
<https://firebase.google.com/docs/database/?hl=es>
- [20] Firebase Storage. <https://firebase.google.com/docs/storage/?hl=es-419>
- [21] Video Streaming Products. Wowza Media System.  
<https://www.wowza.com/products/streaming-engine>
- [22] ¿Qué es Wamp Server? Olga Cirauqui Elorz, Servicio Navarro de Empleo.  
[http://aulasne.navarra.es/pluginfile.php/4847/mod\\_page/content/31/installar\\_wamp.pdf](http://aulasne.navarra.es/pluginfile.php/4847/mod_page/content/31/installar_wamp.pdf)
- [23] NO-IP: Free Dynamic DNS. <https://www.noip.com/>
- [24] Open SSL. <https://www.openssl.org/>
- [25] What is Secure Sockets Layer (SSL)? Margaret Rouse, SearchSecurity.  
<http://searchsecurity.techtarget.com/definition/Secure-Sockets-Layer-SSL>
- [26] The Agile Movement. Agile Methodology. <http://agilemethodology.org/>
- [27] GitLab. <https://about.gitlab.com/>
- [28] Trello. <https://trello.com/>
- [29] What is Git? Atlassian. <https://www.atlassian.com/git/tutorials/what-is-git>
- [30] Versiones de la plataforma. Android Developers.  
<https://developer.android.com/about/dashboards/index.html?hl=es-419>
- [31] Manifiesto de la app. Android Developers.  
<https://developer.android.com/guide/topics/manifest/manifest-intro.html?hl=es-419>



- [32] Gradle Build Tool. <https://gradle.org/>
- [33] GSON, A Java serialization/deserialization library to convert Java Objects into JSON and back. <https://github.com/google/gson>
- [34] Opciones de almacenamiento. Android Developers. <https://developer.android.com/guide/topics/data/data-storage.html?hl=es-419>
- [35] ¿Qué son los metadatos y qué revelan? Sarah Santiago, Open Data Security. <https://opendatasecurity.io/es/que-son-los-metadatos-y-que-revelan/>
- [36] Real-time Transport Protocol. Wikipedia. [https://en.wikipedia.org/wiki/Real-time\\_Transport\\_Protocol](https://en.wikipedia.org/wiki/Real-time_Transport_Protocol)
- [37] H.264 / MPEG-4 AVC. Wikipedia. [https://es.wikipedia.org/wiki/H.264/MPEG-4\\_AVC](https://es.wikipedia.org/wiki/H.264/MPEG-4_AVC)
- [38] H.263 <https://en.wikipedia.org/wiki/H.263>
- [39] Advanced Audio Coding [https://en.wikipedia.org/wiki/Advanced\\_Audio\\_Coding](https://en.wikipedia.org/wiki/Advanced_Audio_Coding)
- [40] Adaptive Multi-Rate audio codec [https://en.wikipedia.org/wiki/Adaptive\\_Multi-Rate\\_audio\\_codec](https://en.wikipedia.org/wiki/Adaptive_Multi-Rate_audio_codec)
- [41] JWPlayer vs VideoJS. SimilarTech. <https://www.similartech.com/compare/jw-player-vs-videojs>