Trabajo de Fin de Grado

E-Dash: Plataforma Android para Dashcams

E-Dash: Android Platform for Dashcams

María del Rocío Rodríguez Morín

La Laguna, 4 de julio de 2015

Dña. **Pino Caballero Gil**, con N.I.F. 43.534.310-Z profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora.

D. **Cándido Caballero Gil**, con N.I.F. 42.201.070-A contratado de investigación 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:

"E-Dash: Plataforma Android para Dashcams"

ha sido realizada bajo su dirección por Dña. **María del Rocío Rodríguez Morín**, con N.I.F. 54.109.765-B.

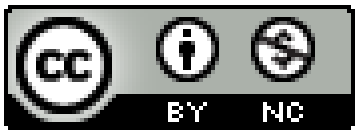
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 4 de julio de 2015.

Agradecimientos

En primer lugar me gustaría agradecerle a Pino y a Cándido el apoyo y la confianza que han depositado en mí desde el principio. También quiero agradecerle al resto del grupo CryptULL, gracias a vuestra ayuda este proyecto ha sido posible.

En segundo lugar, me gustaría dar las gracias a mi familia y amigos, por animarme siempre a continuar con mis estudios y hacer de mí la persona que soy. En especial a mis padres y a mi abuela.

Licencia



Resumen

El principal objetivo de este proyecto ha sido el desarrollo de una plataforma segura y centralizada para gestionar emergencias en carretera, que incluye una aplicación móvil basada en la tecnología subyacente de las dashcams. De esta forma, dicha plataforma permitirá a los servicios de emergencias y otras entidades oficiales actuar con mayor rapidez en caso de accidente, disminuyendo los tiempos de respuesta y aumentando las posibilidades de supervivencia.

Para ello, por una parte se ha desarrollado la aplicación Android llamada E-Dash, que en caso de detectar un accidente del propio vehículo procede a enviar la grabación de la colisión a un servidor centralizado, cifrada usando el algoritmo AES. Además, por otra parte se han desarrollado los servicios web necesarios para ver y consultar de forma casi instantánea diferentes datos de interés incluyendo nombre de la persona accidentada, coordenadas geográficas y video del accidente. La seguridad de dicho servidor está protegida mediante el uso del algoritmo Diffie-Hellman basado en criptografía de curvas elípticas.

Palabras clave: Seguridad, Android, Dashcam, Aplicación Móvil, Servidor.

Abstract

The main objective of this project was the development of a secure and centralized platform for managing emergencies on the road, which includes a mobile application that is based on the underlying technology dashcams. Thus, this platform will allow emergency services and other authorities to act faster when an accident happens, reducing response times and increasing the chances of people survival.

In order to do this, on the one hand, the Android application called E-Dash has been developed, which in case of detecting a vehicle accident, it proceeds to send recording of the collision to a centralized server, encrypted using the AES algorithm. In addition, on the other hand, all web services necessary to view and consult almost instantly different relevant data including name of the injured person, geographic coordinates and video of the accident. The security of the server is protected using the Diffie-Hellman algorithm based on elliptic curve cryptography.

Keywords: Security, Android, Dashcams, Mobile Application, Server.

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

1.1 Motivación

Según diferentes estudios, aproximadamente el 66% de las muertes de los accidentes de tráfico se producen durante los 20 minutos posteriores al accidente. En estas ocasiones es fundamental la rapidez y la coordinación de los servicios de emergencias y la gestión del tráfico. Cada minuto es crucial y disminuir los tiempos de respuesta puede suponer salvarle la vida a una persona.

En la sociedad actual, el uso del smartphone es prácticamente imprescindible, estamos en la era de las tecnologías. De este modo, ¿por qué no aprovechar nuestros teléfonos inteligentes para avisar automáticamente a los servicios de emergencia? Este es el objetivo del proyecto E-Dash.

Figura 1.1. Logotipo de E-Dash

Además, para obtener información más precisa, hacemos en este trabajo uso de una tecnología de origen ruso llamada dashcam.

Las cámaras dashcam se colocan en el salpicadero del coche para grabar los trayectos y en caso de accidente, el usuario dispone de las imágenes que demuestran lo sucedido.

El proyecto E-Dash supone una mejora de las dashcams tradicionales puesto que permite usar teléfonos móviles para la captura de vídeos y enviar dichos vídeos a un servidor donde estarían situados los servicios de emergencias.

Esto implica que el usuario final no necesite adquirir un dispositivo extra e instalarlo en su coche, sino simplemente que le da una nueva utilidad a su Smartphone aumentando la seguridad en su vehículo.

1.2 Tecnología de la aplicación móvil

Antes de comenzar el desarrollo de la aplicación se llevó a cabo la investigación de las diferentes posibilidades de desarrollo y el análisis de los requerimientos de la plataforma.

La primera opción evaluada fue el desarrollo de una aplicación multiplataforma para el móvil, lo que permitiría que E-Dash pudiese ser integrada en los diferentes sistemas operativos para móviles disponibles en el mercado. Sin embargo, estas aplicaciones tienen un gran rendimiento y suelen presentar problemas con la integración de algunos de los componentes nativos de los dispositivos físicos, y esto era necesario para poder emular el comportamiento de las dashcams.

La siguiente opción a evaluar fue el desarrollo nativo en plataformas móviles, donde las destacables son Android e iOS. Estos sistemas operativos son líderes en el mercado de los teléfonos inteligentes. Entre ellos sumaban el 96,3% de la cuota de mercado en el último trimestre del año 2014, siendo el 81,5% el porcentaje correspondiente a los dispositivos que utilizan Android. Por tanto, con la intención de llegar a un mayor número de personas para que el proyecto tenga un mayor impacto social y una mayor utilidad se escogió el desarrollo de la aplicación para Android.

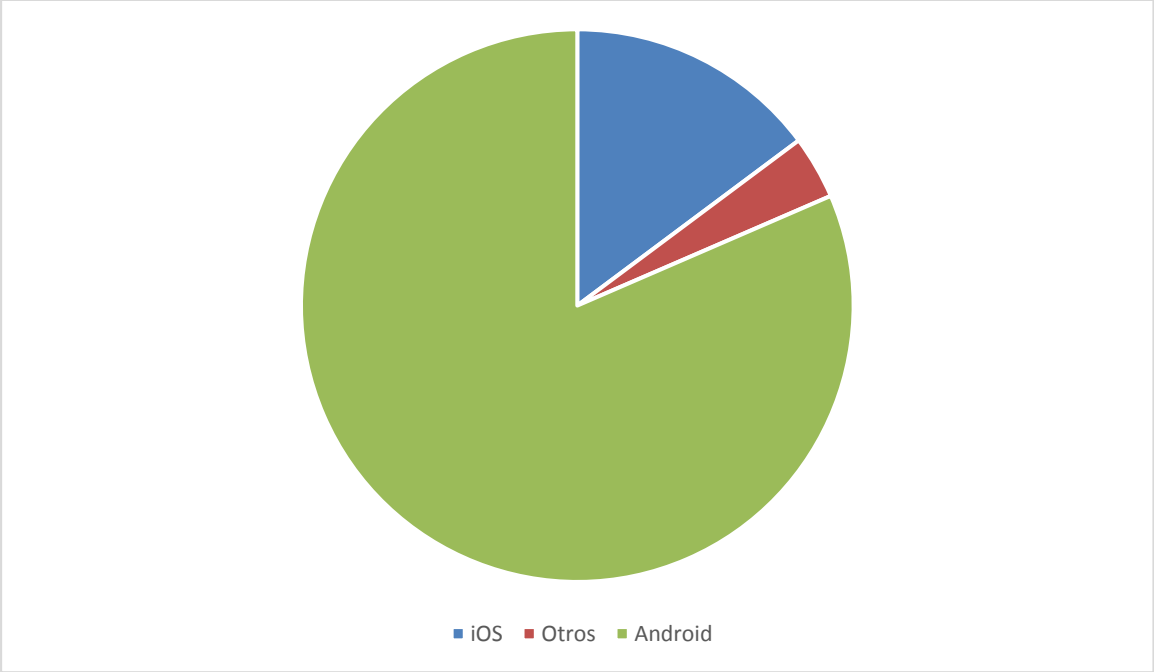


Figura 1.2. Sistemas operativos en el último trimestre de 2014

El SDK (Software Development Kit) de Android, se compone de un conjunto de herramientas de desarrollo entre los que se encuentra un depurador de código, un simulador de smartphones, documentación, ejemplos de código y tutoriales. Las dos plataformas integrales de desarrollo (IDE, Integrated Development Environment) más destacadas son Eclipse y Android Studio. Tras analizar las ventajas y desventajas de cada una, E-Dash se desarrolló en Android Studio por las siguientes razones:

- Utilización de Gradle, herramienta para automatizar la construcción de proyectos.
- Facilidad para construir y generar múltiples versiones de la aplicación.
- Refactorización y autocompletado de código avanzado.
- Vista en tiempo real del renderizado de layouts.

1.3 Tecnología del servidor

Los datos recogidos por E-Dash se envían a un servidor desarrollado con el framework MEAN de JavaScript [2] que da soporte a las aplicaciones web, aumentando la velocidad del desarrollo y facilitándolo considerablemente. Su

nombre proviene de que combina cuatro tecnologías: MongoDB [8], Express [9], AngularJS [10] y Node.js [11].

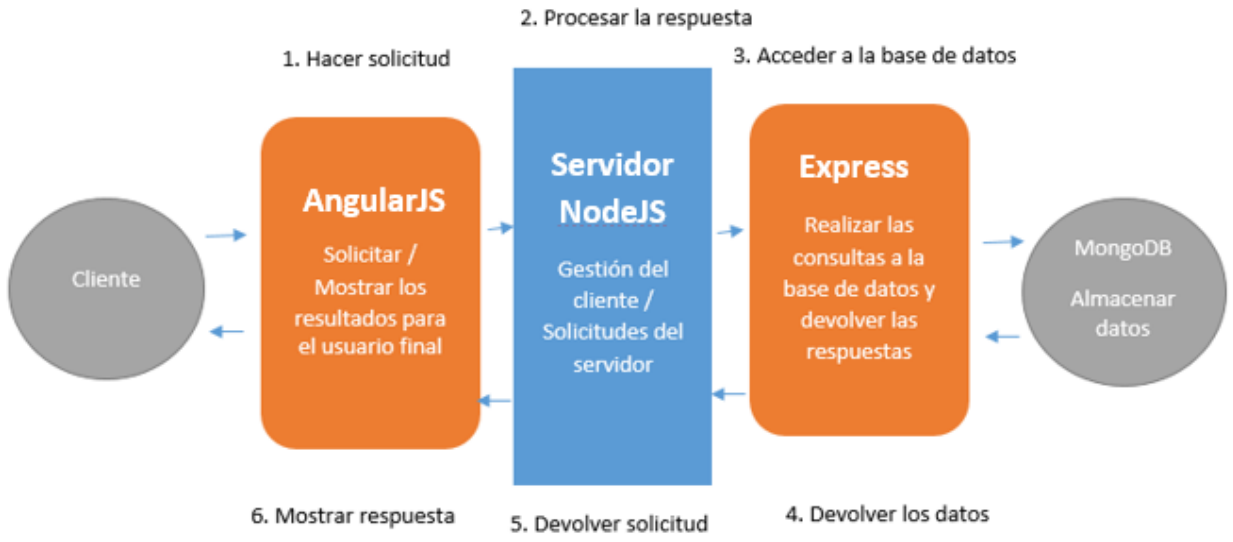


Figura 1.3. Pila de protocolos MEAN

MongoDB es la encargada de gestionar nuestra base de datos no relacional de código abierto. Guarda los documentos en BSON, la versión binaria de JSON (JavaScript Object Notation) para aumentar la velocidad con la que se integra la base de datos. Tiene una alta escalabilidad y una facilidad de integración con el resto de tecnologías del stack MEAN.

Express es un framework a más alto nivel que Node.js que permite crear APIs REST de manera más rápida, facilitando enviar y recibir peticiones HTTP. Esto es una ventaja muy grande a la hora de comunicar la aplicación móvil con el servidor.

La tercera de estas tecnologías, AngularJS, es la responsable del Frontend de la aplicación web, que permitirá a los servicios de emergencia interactuar con los datos de los accidentes de tráfico que tengan lugar. Además, este framework de la parte cliente permite crear Single-Page Applications, que son aplicaciones web que no necesitan recargar la página para actualizarse. Esto permite que en el momento en el que una colisión tiene lugar, el trabajador que esté gestionando

el servidor sea capaz de ver en todo momento la información actual sin tener que recargar la página.

Por último, Node.js se encarga del Backend del servidor, está basado en el motor V8 de JavaScript del navegador Google Chrome y está orientado a eventos no bloqueantes, lo que lo hace muy rápido y robusto.

Una de las principales ventajas de MEAN es que en toda la pila del framework de la aplicación se utiliza el mismo lenguaje de programación, lo cual facilita la comprensión de las diferentes partes de la aplicación web.

1.4 Objetivos

Como objetivo general para el desarrollo de este trabajo se marcó realizar una mejora de las dashcams, añadiendo streaming LTE para trasmitir automáticamente el vídeo en caso de accidente en vehículos civiles. En el período de investigación inicial, se descartó esta tecnología por varios motivos.

El primer motivo es que utilizar dicha tecnología constantemente supone un gran consumo de recursos tanto para el usuario de la aplicación móvil como para el servidor puesto que esto implica enviar en tiempo real los videos de todos los viajes de cada uno de los usuarios.

El segundo motivo es la falta de privacidad que esto implica, ya que la mayoría de los usuarios se negarían a usar una aplicación que va a retransmitir en tiempo real por donde está circulando.

El tercer motivo es que esta retransmisión emitiría el vídeo del trayecto completo en vez de únicamente el vídeo de la colisión, lo que es poco práctico.

La solución que se ha adoptado es grabar intervalos cortos de tiempo y enviar el último vídeo sólo si existe colisión. Esto limita muchísimo más el consumo de datos y de recursos en el móvil y en el servidor, le da al usuario mayor privacidad y facilita la tarea de los servicios de emergencia.

Los objetivos específicos de este trabajo de fin de grado se pueden desglosar en los siguientes:

- Diseño de la aplicación móvil que cumpla con los siguientes requisitos: Realizar autenticación con una red social, grabar vídeos de una

duración determinada, obtener la posición, detectar las colisiones y enviar la información necesaria en caso de accidente.

- Implementación del servidor: Realizar un servidor en Amazon o local que recoja los datos enviados por la aplicación móvil, donde se muestre un listado de las colisiones y además se pueda visualizar el contenido de las imágenes grabadas, así como el mapa con las coordenadas correspondientes.

1.5 Fases del desarrollo

El desarrollo del proyecto se ha llevado a cabo en dos fases claramente diferenciadas.

La primera de ellas fue una fase de investigación en la que se realizó una comparación de las aplicaciones multiplataformas con las aplicaciones nativas de Android. También se estudiaron las diferentes plataformas de desarrollo, contemplando sus ventajas y desventajas, así como las aplicaciones actualmente disponibles relacionadas con el desempeño del proyecto para poder obtener las necesidades de los usuarios.

Una vez concluida la fase de investigación, comenzó la fase de desarrollo, para lo cual la autora tuvo que formarse en las tecnologías seleccionadas (Android y MEAN.js). Puede considerarse que esta fase se dividió en dos partes: el desarrollo de la aplicación móvil y la implementación del servidor, aunque no se realizaron de manera secuencial porque era necesario comprobar que los avances conseguidos en la aplicación se visualizaban de manera correcta en el servidor.

1.6 Estructura de la memoria

La estructura de la memoria está organizada de la siguiente manera:

- Capítulo 2: Describe el estado actual de la tecnología de las dashcam, explicando de manera detallada los principios fundamentales que

utilizan estos dispositivos, así como su funcionamiento y características técnicas.

- Capítulo 3: Contiene los detalles de implementación de cada una de las funcionalidades de la aplicación móvil y las herramientas utilizadas.
- Capítulo 4: Explica la estructura del servidor, las tecnologías utilizadas para el desarrollo y los diferentes módulos que lo componen.
- Capítulo 5: Describe las herramientas utilizadas para implementar la seguridad en el envío de datos entre el terminal y el servidor, así como una descripción detallada del algoritmo Diffie-Hellman elíptico utilizado para el intercambio de claves y AES para el cifrado de los datos.
- Capítulo 6: Recoge el presupuesto detallado del proyecto.
- Capítulo 7: Conclusiones y líneas futuras del proyecto.
- Capítulo 8: Conclusions and open problems.

Capítulo 2.
Preliminares sobre dashcams

2.1 Definición

Como se ha descrito anteriormente, las dashcams son cámaras que se colocan en el parabrisas interior del vehículo y que permiten registrar continuamente la carretera mientras el vehículo está en movimiento. Se puede ajustar por medio de una ventosa o de algún adhesivo a la parte superior del salpicadero o al espejo retrovisor con una montura especial.

Figura 2.1. Ejemplo de dispositivo dashcam

Esta tecnología está bastante extendida en Rusia, donde son utilizadas como un mecanismo de vigilancia inversa. Su función principal es actuar como evidencias visuales en caso de accidente y como medida de prevención de la corrupción policial y el fraude de seguros. Una gran cantidad de vídeos provenientes de estos dispositivos han sido compartidos en diferentes plataformas como YouTube o Facebook y muestran todo tipo de circunstancias ocurridas durante el trayecto de los conductores.

Existen diferentes tipos y modelos de dashcams, desde grabadoras de vídeo básicas hasta algunas más sofisticadas capaces de medir diferentes parámetros como el tiempo, la fecha, la velocidad, la fuerza G y la ubicación.

Estos dispositivos están generando una gran popularidad en diferentes zonas de Asia, Europa, Australia y Norteamérica, sin embargo, en algunos países de Europa, como Austria, Suiza y Alemania, estos dispositivos generan gran polémica, llegando a estar prohibido su uso por ley en algunos de ellos.

En general, nos encontramos con tres tipos de dashcam:

- Dashcam frontal o de un canal. Normalmente se colocan en el parabrisas delantero o en el salpicadero mirando hacia delante.
- Dashcam dual o de dos canales. Además de la cámara frontal, también hay una cámara secundaria que se coloca en la parte trasera del vehículo para proporcionar una cobertura de visión delantera y trasera.
- Dashcam de cuatro canales. A parte, de las cámaras delantera y trasera, se colocarían dos cámaras más, situadas a ambos lados del vehículo.

Las cámaras de más de un canal graban y almacenan simultáneamente los videos en una sola tarjeta de memoria centralizada, aunque algunos modelos nuevos son capaces de guardar estos archivos de manera individual. A menudo, grandes empresas con vehículos comerciales tienden a usar dashcams de múltiples canales.

2.2 Funcionamiento

Las dashcams funcionan más o menos como cualquier otra cámara de video, exceptuando la característica diseñada para la grabación en bucle. Esta grabación es almacenada en una tarjeta SD (Secure Digital) y una vez que la capacidad máxima es alcanzada se sobrescriben las imágenes de mayor antigüedad. Estos dispositivos poseen un sensor de gravedad capaz de detectar cambios bruscos en la aceleración, reconociendo de este modo una colisión. En el instante en el que esto ocurre se protegen las imágenes.

La energía que precisan las cámaras para funcionar se puede obtener de dos maneras, enchufándola en el adaptador de 12V que traen los coches por defecto,

o también se puede cablear directamente a la batería del vehículo, proporcionando la ventaja de liberar el conector de 12v del vehículo para otros posibles usos.

El proceso de grabación por lo general está automatizado y se inicia al segundo o minuto de encender el vehículo, y lo mismo pasa al apagar el vehículo. La mayoría de los modelos de dashcam graban segmentos en archivos de 1, 3 o 5 minutos para permitir un medio rápido y eficaz de acceso cuando se necesita reproducir los archivos multimedia.

2.3 Características técnicas

En general, casi todos los modelos de dashcam contendrán las siguientes especificaciones:

- **Resolución de vídeo (VGA, HD, FullHD):**La calidad del vídeo dependerá de la cámara. Ahora mismo, hay una amplia variedad de modelos disponibles en el mercado, desde aquellas que graban en resolución VGA(640x480)y en blanco y negro, hasta otras que graban hasta 1080p Full HD.
- **Ángulos de visión amplios:** Poseen habitualmente entre 120 y 180 grados.
- **GPS (Global Positioning System):** La mayoría de las dashcams vienen con un receptor GPS integrado o con un módulo adicional que permite a la cámara capturar la velocidad y los detalles de localización.
- **Grabación en bucle:** Como ya se ha mencionado anteriormente, las dashcams graban en ficheros de intervalos cortos. Una vez que la memoria se acaba, esta opción permite sobrescribir los primeros archivos almacenados y continuar con el viaje.
- **Sensor-G (Sensor de gravedad):** Es una de las características más importantes de las dashcams y hablaremos más extensamente acerca de ella, en el capítulo referente al desarrollo de la aplicación móvil. Este sensor se encarga de detectar si un impacto o un movimiento irregular del vehículo tiene lugar, el sensor se activará y protegerá el vídeo. De esta manera el bucle no afectará a las imágenes de la colisión.

2.4 Aspectos legales

El uso de las dashcams ha generado bastantes dudas en lo relativo al tratamiento de las grabaciones recogidas. En ejercicio de la competencia que le atribuye el artículo 37.1.c) de la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal, la Agencia Española de Protección de Datos dio a luz la Instrucción 1/2006. Mediante la misma la AEPD [16] busca adecuar los tratamientos de imágenes a los principios de dicha Ley Orgánica y garantizar de esta manera los derechos de las personas cuyas imágenes son tratadas por medio de procedimientos como la grabación mediante dashcam.

La AEPD considera como dato de carácter personal:

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

En los vídeos de dashcam se enfoca la vía pública, por la que transitan personas y vehículos. Aún en el caso de que se consiguiera que no salgan personas, hay que tener en cuenta que *"se considerará identificable toda persona cuya identidad pueda determinarse, directa o indirectamente, en particular mediante un número de identificación o uno o varios elementos específicos, característicos de su identidad física, fisiológica, psíquica, económica, cultural o social"*. Considerando lo anterior, la matrícula de un vehículo es un dato de carácter personal.

Para poder utilizar estas imágenes deberíamos tener el consentimiento de todas las personas que salen en el vídeo así como de todos los propietarios de los coches que circulen o estén estacionados. Además, deberíamos anunciar a las personas de que están siendo videovigiladas. Por lo tanto, concluimos que a día de hoy, resulta imposible y está totalmente prohibido el uso de estos dispositivos siempre y cuando se grabe el exterior del vehículo.

Sería necesario un cambio de ley que al menos permita estos dispositivos para mejorar la seguridad vial, como es el caso de este proyecto. Siempre respetando la integridad y los derechos de los viandantes, restringiendo el uso de las imágenes a servicios de emergencia. Gracias a este cambio se podría multiplicar

el control en las carreteras, actuar con una mayor rapidez ante un accidente, esclarecer las causas de los mismos con mayor celeridad o reducir los niveles de fraude como ya se ha hecho en otros países.

Capítulo 3. Desarrollo de la aplicación móvil

La aplicación móvil [3] ha sido desarrollada con un SDK mínimo de la API 16, más conocida como JellyBean. Esto implica la compatibilidad con el 87,9% de los dispositivos Android disponibles actualmente. Además, las pruebas de la aplicación se han realizado sobre la API 21 o Lollipop, que es la última versión del sistema operativo para garantizar que los bugs que tienen lugar en las nuevas versiones sean corregidos.

3.1 Geolocalización

Para que E-Dash tenga aplicación en el mundo real y los servicios de emergencia puedan llegar lo antes posible al lugar del accidente, resulta imprescindible que exista un mecanismo de localización [5].

Debido a que los accidentes tendrán lugar en espacios exteriores, la opción más viable y precisa para lograr las coordenadas es el Sistema Global de Posicionamiento o GPS (Global Positioning System).

Android permite la posibilidad de realizar esto con la clase *LocationManager* con lo cual podemos averiguar y obtener los cambios de posición obtenidos por cualquiera de nuestras aplicaciones.

Para el acceso a las coordenadas GPS en E-Dash se ha definido una clase java que crea una instancia del *LocationManager* y que contendrá los servicios del sistema que hacen referencia a la localización.

```
LocationManager locationManager;
String context = Context.LOCATION_SERVICE;
```

```
locationManager =
    (LocationManager)mContext.getSystemService(context);
String provider = LocationManager.GPS_PROVIDER;
Location location = locationManager.getLastKnownLocation(provider);
return location;
```

Posteriormente se guardará el proveedor de GPS en una variable y se pedirá la última posición conocida. Esta función devolverá un vector de tipo String cuya primera coordenada será la coordenada longitudinal y la segunda será la latitud. Si no hay suficiente señal como para obtener las coordenadas, la llamada a esta función devuelve el String "Sin señal" para cada una de las coordenadas.

```
String[] coordenadas = {"", ""};
if (location != null){
    coordenadas[0] = String.valueOf(location.getLatitude());
    coordenadas[1] = String.valueOf(location.getLongitude());
}else{
    coordenadas[0] = "Sin señal";
    coordenadas[1] = "Sin señal";
}
return coordenadas;
```

Acceder a la localización requiere solicitar permisos, así que hay que especificar el permiso ACCESS_FINE_LOCATION en el manifiesto del proyecto.

3.2 Integración con el SDK de Facebook

El primer paso que se llevó a cabo durante el desarrollo del proyecto fue integrar una plataforma de red social que nos diera acceso a algunos datos que nos permitiesen identificar al usuario que ha tenido una colisión.

El SDK de Facebook [4] para Android permite a los usuarios iniciar sesión e incluso gestionar permisos para que la aplicación realice determinadas acciones en Facebook. En este caso, se utiliza para el registro de los usuarios y para enviar el nombre de usuario y otros datos de interés al servidor si un accidente tuviese lugar.

Para utilizar el SDK fue necesario configurar de manera correcta la aplicación E-Dash en el sitio web de desarrolladores de Facebook, como se observa en la Figura **3.1.** Donde cada aplicación debe tener un identificador, la versión de API utilizada y la firma de la aplicación.

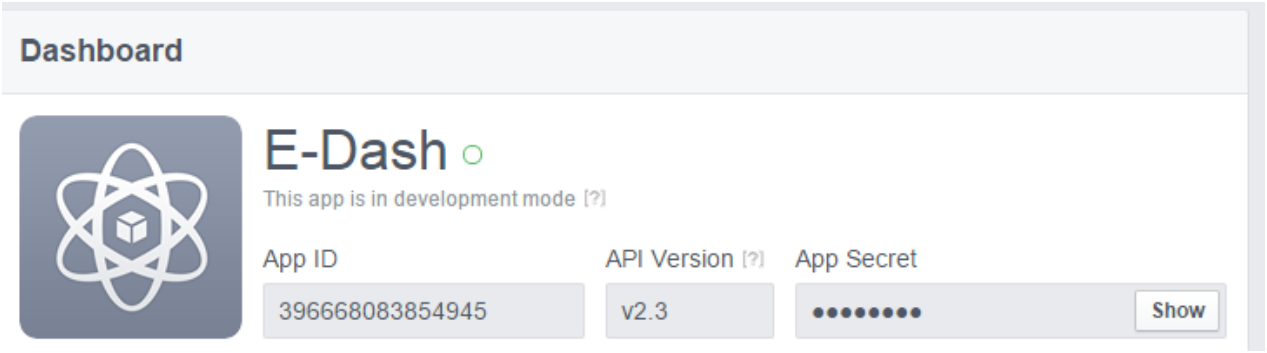


Figura 3.1. Obtención del ID de la aplicación

La creación de una aplicación Android que integre Facebook requiere que se especifique una firma digital en la configuración. Para realizar esta tarea se utilizó la herramienta keytool proporcionada por Java Runtime Environment (JRE).

Una vez configurado, se desarrolló la actividad Android que se lanza en primer lugar cuando un usuario ejecuta la aplicación. Tal y como se muestra en la Figura **3.2**, en esa actividad el usuario simplemente tiene que hacer click en el botón de login e introducir los datos referentes a su cuenta de Facebook.

Si los datos son incorrectos, saldrá un mensaje de error y el usuario tendrá que volver a introducir los datos. En caso contrario, la grabación del trayecto del viaje comienza en la siguiente actividad.

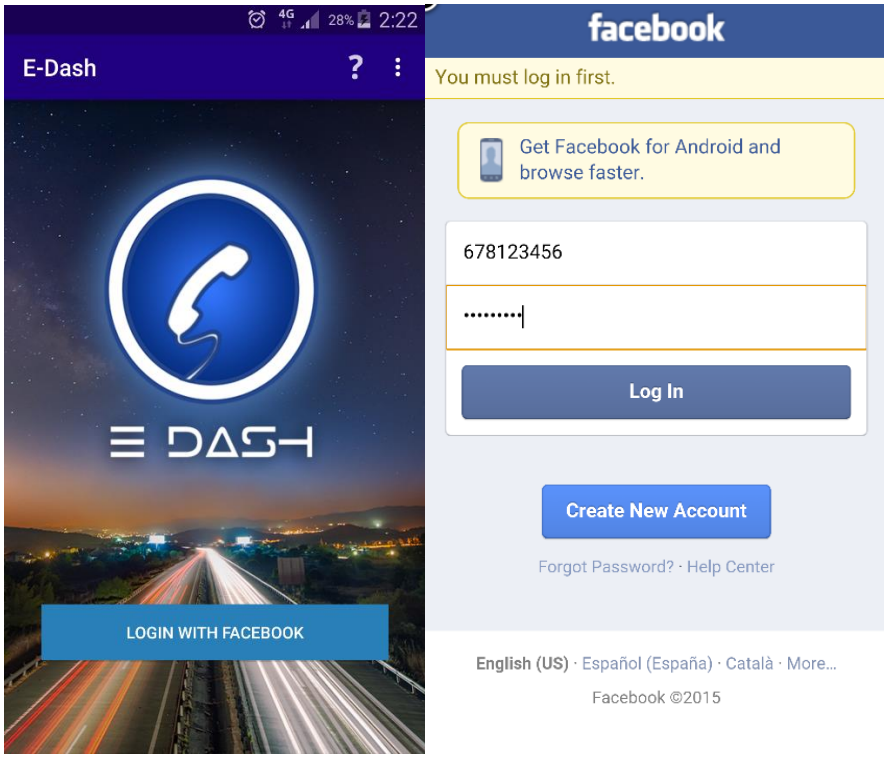


Figura 3.2. Inicio de sesión con Facebook

En la clase java de la actividad, se llama a la función *FacebookSdk.initialize* para inicializar el SDK, y el método *create* de clase *CallbackManager.Factory* para manejar las respuestas de la autenticación de usuarios. Finalmente se llama al método *onActivityResult* del *callbackManager* definido previamente para pasarle los resultados del login al *LoginManager*.

```
callbackManager = CallbackManager.Factory.create();
LoginManager.getInstance().registerCallback(callbackManager,
                                            facebookLoginResultCallback);
    facebookAccessToken = AccessToken.getCurrentAccessToken();
```

Si no se producen errores durante el registro, se ejecuta un Intent encargado de iniciar la siguiente actividad, a la que se le pasan como parámetros los siguientes datos obtenidos desde Facebook: id del usuario, email y nombre completo.

3.3 Grabación de vídeo

En muchos aspectos, construir una aplicación de grabación de vídeo es como implementar una aplicación que captura imágenes combinada con una que captura audio. En primer lugar, hay que establecer un *SurfaceView* para la previsualización de la cámara.

En segundo lugar, se debe elegir entre alguna de las múltiples formas de capturar vídeo [1]. Se está convirtiendo en cliché que la manera más fácil y rápida de realizar una función en Android es usar una aplicación existente que puede ser ejecutada por un Intent, grabar un vídeo no es una excepción. En el caso de E-Dash, es necesario grabar un número indefinido de vídeos de manera secuencial y establecer determinados parámetros como la duración y calidad de los mismos, por ello, no es posible la realización de esta tarea de la manera explicada anteriormente.

Sin embargo, existen otras formas de capturar vídeo. Para ello, el SDK de Android incluye una clase llamada *MediaRecorder* que nos permite crear nuestra propia funcionalidad de grabación. Haciendo esto, se gana mucha flexibilidad, como por ejemplo, poder ajustar la duración de las grabaciones.

Para comenzar la captura debemos seguir una serie de pasos ya que esta clase es una máquina de estados como se puede observar en la Figura **3.3,** y ciertos métodos sólo pueden ser llamados cuando el estado actual es el correcto.

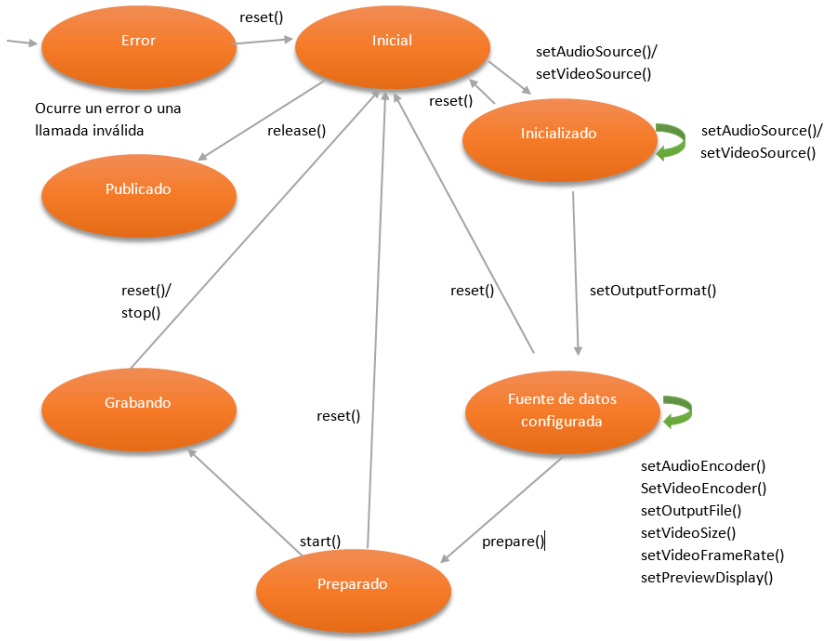


Figura 3.3. Diagrama de estados de *MediaRecorder*

Los métodos más importantes utilizados durante la implementación son los siguientes:

- Inicialización: Una vez que los parámetros de configuración han sido establecidos, se utiliza el método *prepare*. La utilización de esta función es obligatoria e inicializa todas las funciones internas que hacen que el objeto del tipo *MediaRecorder* esté listo para grabar.

  recorder.prepare();

- Comenzando a grabar: Para iniciar la grabación se llama al método *start*. Es importante señalar que estos métodos tienen que llamarse de manera secuencial y que no podemos empezar a grabar sin haber preparado la instancia del objeto.

  recorder.start();

- Parando la grabación: Después de empezar la grabación, el método *stop* es llamado para detener la grabación.

  recorder.stop();

- Liberación de recursos: Finalmente, una vez que hemos terminado con *MediaRecorder*, debemos llamar al método *release* para liberar los recursos. Esto es importante porque dos aplicaciones no pueden acceder

de forma simultanea a los recursos subyacentes, como al hardware de la cámara y el del micrófono.

```
recorder.release();
```

Comenzando por el primer estado, instanciamos el objeto recorder. Una vez hecho esto, pasamos al siguiente estado en el que se configuran las fuentes de los datos. Para fijar la fuente del audio, se llama al método *setAudioSource* pasándole la constante *MediaRecorder.AudioSource.MIC* para especificar que queremos que se utilice el micrófono del dispositivo. De manera análoga se realiza el mismo procedimiento para el vídeo con la constante *MediaRecorder.VideoSource.CAMERA*.

Adicionalmente, dos métodos son llamados antes de preparar al objeto de tipo *MediaRecorder* para grabar: *setOutputFile* y *setOutputFormat*. El primero de ellos nos permite especificar en qué fichero de salida se va a guardar el vídeo y el segundo sirve para establecer el formato del vídeo resultante. Los formatos posibles de clase *OutputFortmat* son los siguientes:

- DEFAULT: Especifica el formato de salida por defecto, este puede variar dependiendo del dispositivo físico.
- RAW_AMR: Esta configuración sólo funciona para audio.
- THREE_GPP: El audio y el vídeo capturado se guardarán en un archivo 3GP.
- MPEG_4: Especifica que el audio y el vídeo serán guardados en un archivo con formato MPEG-4. Los archivos con este formato son ampliamente utilizados por diversas tecnologías y dispositivos razón por la cuál será el formato de salida de E-Dash.

Después de establecer el formato de salida, se especifican los codificadores de video usando la función *setVideoEncoder* pasándole la constante correspondiente. Los posibles valores que puede tomar son:

- DEFAULT: Esta configuración varía dependiendo del dispositivo. En la mayoría de las ocasiones el valor de esta constante será H.263 ya que es el único códec que debe soportar obligatoriamente los dispositivos Android.

- H263: Es un códec publicado en 1995 y fue desarrollado específicamente para vídeos que tienen una tasa de transmisión de bits por segundo baja. Se convirtió en la base para muchas tecnologías durante los inicios de Internet.
- H264: Fue publicado en 2003 y es usado en diversas tecnologías desde BlueRay hasta Flash. La mayoría de los dispositivos Android soportan este códec para reproducción multimedia pero un grupo muy pequeño lo usan para la codificación de vídeo.
- MPEG_4_SP: Publicado en 1999 y desarrollado para ser usado por tecnologías que requieren video de un ratio de bits bajo sin requerir mucha potencia del procesador.

Para la grabación de vídeo de este proyecto se ha utilizado los valores de encoders por defecto.

Desde Android 2.2 (API 8), la clase *MediaRecorder* tiene un método, llamado setProfile, el cuál toma una instancia de *CamcorderProfile*. Esto nos permite definir la calidad de los vídeos usando las constantes *QUALITY_HIGH* o *QUALITY_LOW*. En el caso de E-Dash, como la rapidez a la hora de enviar los datos es más importante que la calidad en la que se reproduzcan las imágenes se ha elegido la segunda opción, lo cual establece los siguientes valores:

- Ratio de bits del audio: 12,200 bits por segundo
- Canales de audio: 1
- Codec del audio: AMR-NB
- Tasa de muestreo del audio: 8000 Hz
- Duración: 30 segundos
- Formato del archivo: 3GPP
- Ratio de bits del vídeo: 256,000 bits por segundo
- Codec del vídeo: 3
- Anchura del vídeo: 176 píxeles
- Altura del vídeo: 144 píxeles
- Ratio del frame del vídeo: 15 frames por segundo

Estos ajustes pueden variar dependiendo de las características del dispositivo. Para establecer estos valores, en la clase *Record* del proyecto, existe una función encargada de inicializar el objeto de clase *MediaRecorder*. En resumen, la configuración elegida para la grabación es la siguiente:

```
recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
recorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
recorder.setMaxDuration(MAX_DURATION);
CamcorderProfile cpHigh =
    CamcorderProfile.get(CamcorderProfile.QUALITY_LOW);
recorder.setProfile(cpHigh);
```

La aplicación crea una carpeta llamada EDash dentro de la carpeta de almacenamiento externo conocida como DCIM, donde se almacenarán los vídeos de las colisiones.

Para que sea posible guardar los archivos resultantes en el almacenamiento externo es necesario solicitar los siguientes permisos:

- RECORD_AUDIO
- CAMERA
- WRITE_EXTERNAL_STORAGE

Los resultados obtenidos de la implementación de grabación se pueden observar en la figura **3.4**, que refleja el layout de la actividad de Android que se corresponde con la emulación de la grabación de vídeos de una dashcam. Se compone de tres elementos principales: la *actionbar* característica de las aplicaciones móviles, la previsualización de las imágenes que recoge la cámara en cada momento y un botón que permite al usuario terminar de grabar cuando finaliza su trayecto.

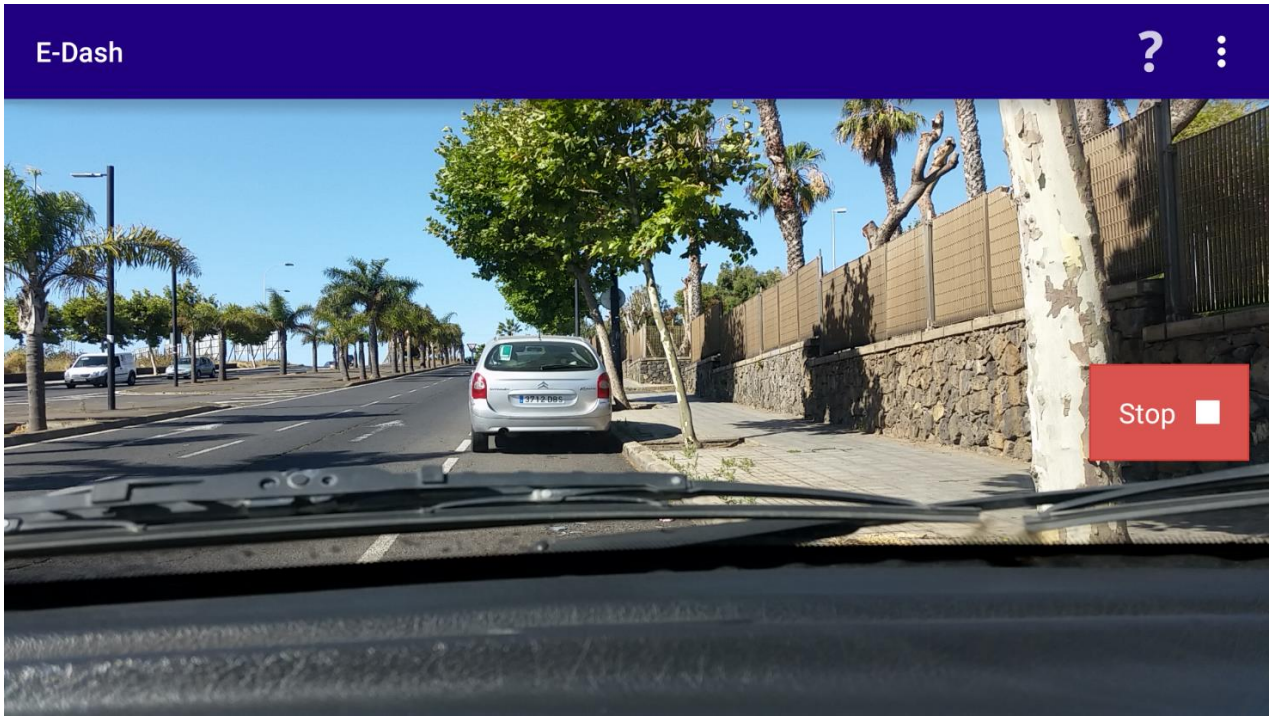


Figura 3.4. Layout de la actividad de grabación

3.4 Detección de colisiones

Los teléfonos inteligentes disponen de diferentes sensores que nos permiten recoger información del entorno. Para detectar una colisión es necesario hacer uso del acelerómetro, el sensor encargado de realizar mediciones de la aceleración aplicada al dispositivo, incluyendo la fuerza de la gravedad.

Conceptualmente, un sensor de aceleración determina la aceleración que es aplicada a un dispositivo (Ad) dividido entre la medición de las fuerzas que son aplicadas sobre el propio sensor (Fs) usando la siguiente relación:

$$Ad = - \sum Fs \ / \ masa$$

Sin embargo, la fuerza de la gravedad está siempre influenciando la medida de la aceleración de acuerdo con la expresión anterior, ya que esta fuerza actúa sobre el sensor y, por tanto, se incluye en el sumatorio.

Por todo esto, cuando el dispositivo reposa sobre una superficie y está en reposo, el acelerómetro hace una lectura de g de 9,81 m/s2. De manera análoga, cuando el dispositivo está en caída libre la aceleración detectada debería ser de 9,81 m/s2 mientras que la magnitud registrada es de 0 m/s2. Por ello, para

realizar una medida real de la aceleración del dispositivo, la contribución de la fuerza de la gravedad debe ser eliminada de los datos recogidos por el acelerómetro. Esto se consigue creando un filtro para aislar esta fuerza.

Teniendo en cuenta todo esto, la detección de colisiones en este proyecto utiliza el acelerómetro para detectar un cambio brusco en la aceleración superior a 2,7 G; ya que cualquier fuerza que supere este valor está considerada colisión. Cuando esto sucede se muestra un cuadro de diálogo como el que se muestra en la figura **3.5**. En caso de que el usuario indique que no ha habido colisión la grabación continúa hasta que el botón de Stop sea pulsado. Si el usuario da una respuesta afirmativa se procede a enviar los datos al servidor. En caso de accidente se podría dar el caso de que el pasajero estuviera inconsciente, por ello si se detecta un movimiento brusco y no se obtiene respuesta también se procede al envío de la colisión. En este caso, le correspondería a los servicios de emergencia determinar si se ha producido un impacto o ha sido una falsa alarma.

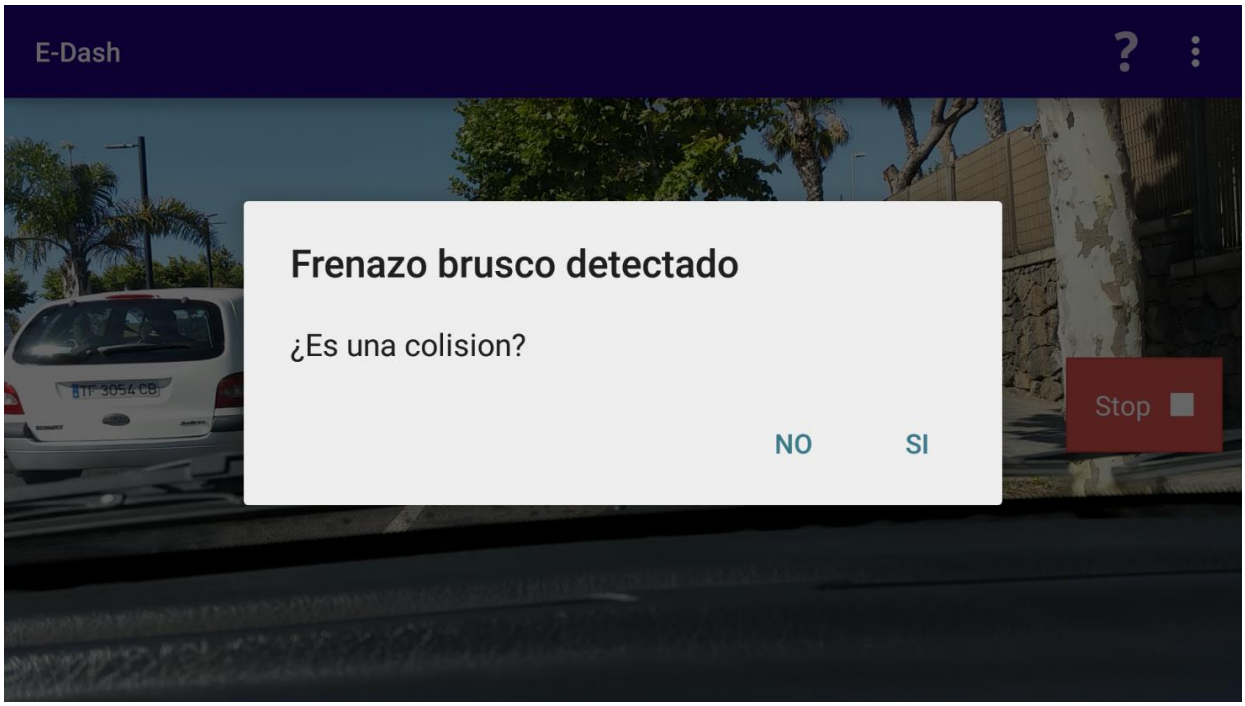


Figura 3.5. Layout de la detección de colisiones

Si es necesario enviar los datos al servidor la función *startToSend* desactiva el hilo encargado de detectar colisiones y lanza la actividad Colision, donde se

implementan los servicios web encargados de comunicarse con el servidor. En esta función se transmiten los datos a la nueva actividad de la siguiente manera:

```
public void startToSend() {
    Detecciones.desactivar();
    Intent intent = new Intent(Record.this, Colision.class);
    intent.putExtra("user_id", id);
    intent.putExtra("user_email", email);
    intent.putExtra("user_name", name);
    intent.putExtra("video_name", videoName);
    intent.putExtra("file", fileUri.getPath());
    startActivity(intent);
}
```

De manera análoga, si no se detecta colisión, otra función es la encargada de lanzar la actividad que anuncia el fin del trayecto de manera satisfactoria.

3.5 Servicios web

Una vez que hemos detectado y grabado la colisión, el siguiente paso es enviar la información al servidor, para ello se utilizan, entre otros muchos usos, los servicios web. Esta tecnología permite a distintas aplicaciones compartir información y ficheros a través de diferentes estándares y protocolos.

En el desarrollo de aplicaciones Android existen diversas maneras de establecer la comunicación entre una aplicación móvil y un servidor. No obstante, en estos métodos el proceso de establecimiento de la conexión y descarga es bastante complicado.

Para el desarrollo de estos servicios web en E-Dash se ha utilizado *Volley*, una librería de Google que proporciona una solución de networking que pretende abstraer al desarrollador de los problemas descritos anteriormente.

Antes de realizar el servicio web, se importó *Volley* a través del repositorio clon de *Maven*, de manera que se actualice automáticamente cuando se realicen cambios en el repositorio oficial añadiéndolo a las dependencias del proyecto:

compile 'com.mcxiaoke.volley:library:1.0.+'

La implementación de los servicios web usando *Volley* se realizaron mediante la creación de una cola de peticiones a la que se le pasan como parámetros objetos de tipo *Request*. Esta cola gestiona hilos para ejecutar las operaciones de envío y recepción, las lecturas y escrituras de la caché y el análisis de las respuestas. Las solicitudes hacen el análisis de las respuestas y *Volley* se encarga de devolver el resultado al hilo principal para su posterior procesado.

E-Dash implementa una petición POST al servidor para enviar los datos de usuario, la localización y el vídeo. Cuando se detecta un accidente, se ejecuta la petición y el objeto de tipo Request es añadido a la cola de peticiones con el método *add*. Una vez hecho esto, se le da servicio a la petición y su respuesta es gestionada.

Cuando el método *add* es llamado, *Volley* ejecuta un hilo de procesamiento de caché y un grupo de hilos de subprocesos de red. Cuando se añade una petición a la cola, es recogida por el hilo de caché y si la petición no puede ser resuelta, la petición es trasladada a la cola de red. El primer hilo disponible atiende la petición y realiza la petición HTTP, escribe la respuesta en caché y la devuelve al hilo principal. En la Figura **3.6** se puede observar el ciclo de vida de una petición.

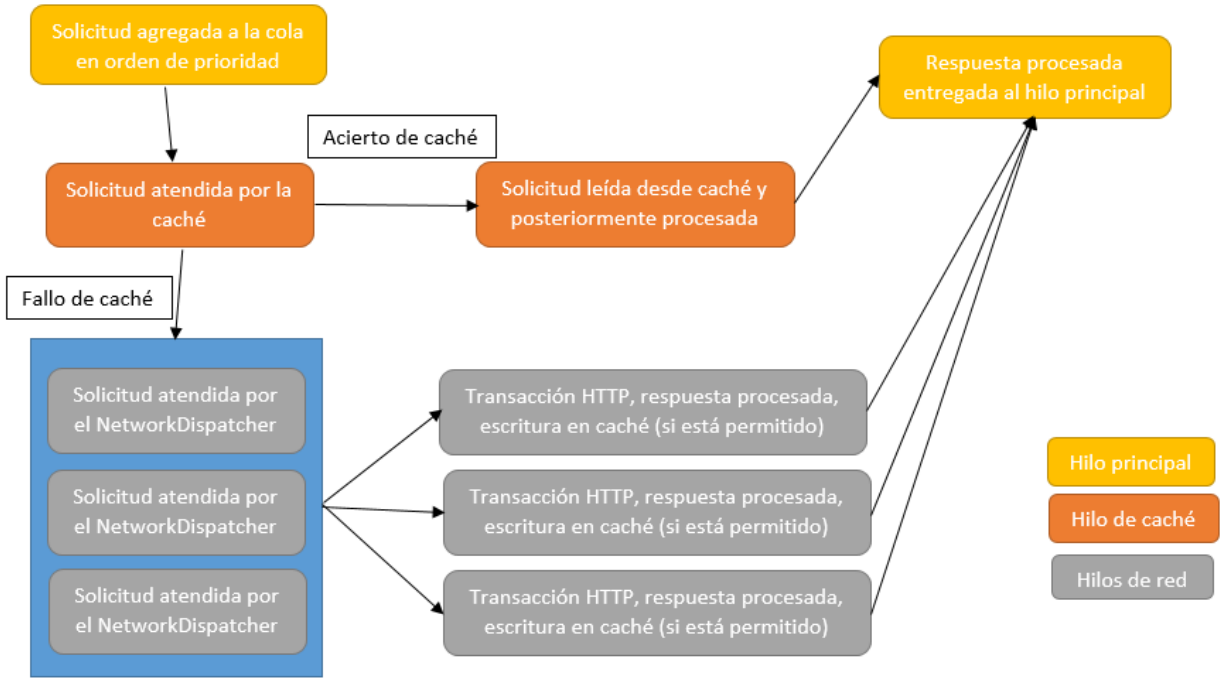


Figura 3.6.Ciclo de vida de una petición HTTP

Aunque *Volley* soporta diferentes tipos de peticiones, por cuestiones del servidor y facilidad a la hora de interpretar la información, durante la realización del proyecto se han utilizado peticiones que envían un objeto de tipo JSON para lo cual es necesario crear una instancia de *JsonObjectRequest* y unas subclases de *JSONObject* especificando la URL y obteniendo como respuesta un objeto JSON.

JSON son las siglas de JavaScript Object Notation y aunque fue diseñado originalmente para ser usado en JavaScript, es independiente del lenguaje. Es ampliamente utilizado porque su implementación es relativamente fácil, es más ligero que otros formatos, más compacto y más fácil de procesar por los ordenadores que otros lenguajes como xml.

El objeto JSON construido en el servicio web contiene los siguientes datos:

- Id de usuario
- Email del usuario
- Nombre de usuario
- Coordenada longitudinal del accidente
- Coordenada latitudinal del accidente
- Nombre del vídeo

- Vídeo del accidente

Una vez que el objeto JSON es creado con los datos anteriores, se procede a establecer la política de reintentos y a añadir la solicitud a la cola de la siguiente manera:

```
int socketTimeout = 30000;
RetryPolicy policy = New DefaultRetryPolicy(socketTimeout,
                        DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
                        DefaultRetryPolicy.DEFAULT_BACKOFF_MULT);
postRequest.setRetryPolicy(policy);
queue.add(postRequest);
```

La implementación de estos servicios utiliza como canal la conexión a internet y por tanto, Android requiere añadir al manifiesto del proyecto el permiso de INTERNET.

Capítulo 4.
Desarrollo del servidor

4.1 Modelo Vista Controlador (MVC)

La aplicación web desarrollada en este proyecto está dividida en dos partes, el lado del servidor y el del cliente. Ambas siguen el patrón Modelo-Vista-Controlador.

- Modelos: Representan conocimiento. Son objetos que almacenan la información y permiten su manipulación, típicamente conocidos como bases de datos. En este proyecto este papel lo desempeña la base de datos implementada sobre la tecnología de *MongoDB* [8] utilizando *mongoose* como middleware para describir el esquema. En él se almacena la información de las colisiones que se reciben desde los dispositivos móviles.
- Vistas: Son las representaciones visuales del modelo. Una vista esta adjunta a su correspondiente modelo y obtiene, a través de consultas a la base de datos, la información necesaria para realizar la representación. Para que esta comunicación sea posible, las vistas deben conocer la terminología del modelo. Esto implica que la vista debe conocer la semántica de los atributos que el modelo representa. En el proyecto vienen representadas por cada una de las páginas web que visita el usuario para poder registrarse o iniciar sesión, listar u observar las colisiones.
- Controladores: Son el enlace entre el usuario y el sistema. Deciden el significado de las acciones realizadas por el usuario, los cambios que se han de realizar sobre el modelo como resultado de esos eventos y cual es la vista resultante que debe ser renderizada.

La Figura **4.1** ilustra como se produce el flujo de información entre cada una de estas partes.

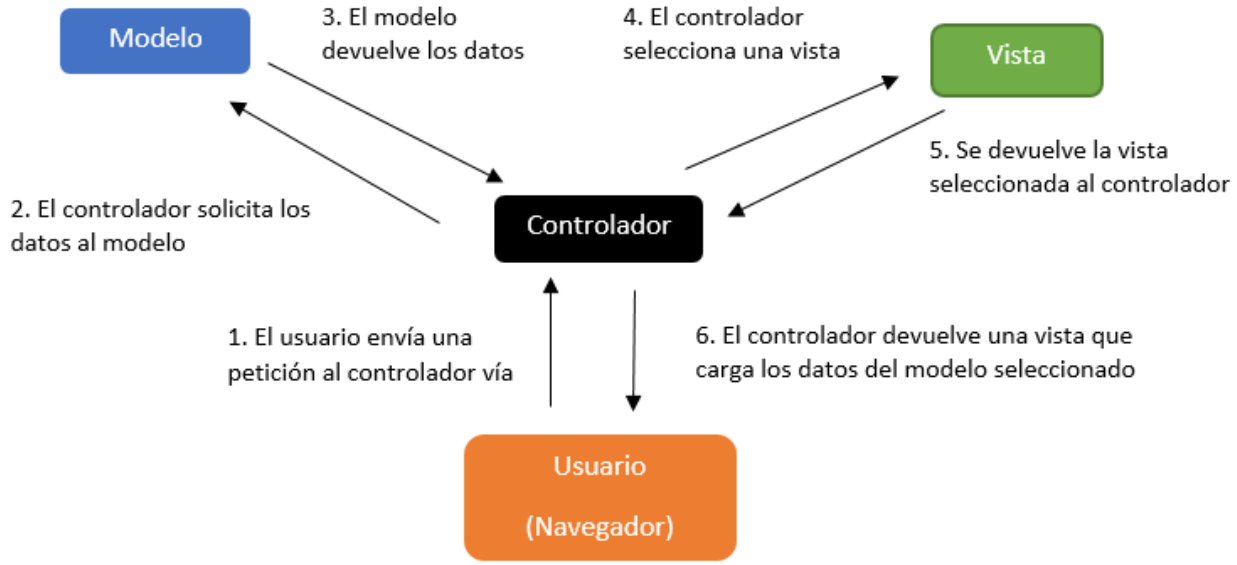


Figura 4.1. Diagrama del MVC

4.2 Estructura del servidor

Existen tres módulos en el proyecto. El primero de ellos es el módulo core y es el encargado de establecer el layout de la aplicación web, así como la ruta raíz del proyecto. El segundo módulo es el de los usuarios, encargado de gestionar los registros y sesiones de los trabajadores de los servicios de emergencia. El último módulo hace referencia a las colisiones. En él están definidos los atributos de los accidentes que son enviados desde el teléfono. Además, se realizan acciones como listar colisiones, mostrarlas o borrarlas. Se hablará detenidamente de estos dos últimos en los apartados **4.3** y **4.4** de la presente memoria.

La aplicación web de este proyecto está dividida en dos grandes partes**:** el lado del cliente y el lado del servidor. En el *backend* se encuentran diferentes carpetas que hacen referencia las distintas tecnologías que componen el framework *MEAN*: los modelos, las rutas, las vistas y los controladores. Los ficheros más importantes del lado del servidor son:

- controllers: En ella se implementa la lógica de negocio del backend a través de aplicaciones Express.
- models: Contiene los modelos del backend usando Mongoose.
- routes: En esta se almacenan los archivos e configuración de rutas del servidor.
- views: Almacena las vistas del *backend*. Con el uso de Angular JS las vistas del backend pierden importancia pero aún así es necesario un motor de plantillas para renderizar las páginas principales, como el index o las páginas de errores.
- config: Contiene los archivos de configuración de la aplicación.
- env: Almacena los archivos de configuración cargados por el archivo *config.js* según el entorno actual sobre el que se está ejecutando la aplicación.
- strategies: Contiene las estrategias de configuración de inició de sesión con las diferentes redes sociales cargadas por el archivo passport.js.
- config.js: Es el fichero encargado de cargar la configuración adecuada en función del entorno que se esté utilizando.
- express.js: Fichero de configuración de Express que inicializa y configura la aplicación express.

En el contenido de la carpeta *public* está almacenado el *frontend*. A diferencia del *backend*, aquí los ficheros están agrupados por funcionalidad. De esta manera, cada uno de los módulos que se añadan al proyecto tendrá sus propios archivos. Destacan por su importancia los siguientes archivos:

- dist: La carpeta de distribución es donde los archivos CSS comprimidos y los ficheros JavaScript son almacenados.
- modules: Contiene las aplicaciones modulares exportadas de AngularJS necesarios para realizar funciones específicas. Como por ejemplo el módulo para importar Bootstrap.
- application.js: Es el archivo principal de la aplicación AngularJS, se encarga de aplicar Bootstrap y adjuntarle los módulos correctos.

Además, existen archivos de aplicación almacenados en el directorio raíz:

- server.js: Es el fichero principal de la aplicación web, donde se inicializa la aplicación Node.js.
- bower.json: Archivo de definición de *bower* donde se configuran los componentes del *frontend* que son necesarios.
- gruntfile.js: Archivo de definición de*grunt*, encargado de ejecutar el servidor.
- package.json: Archivo de definición *npm* [13], donde se configuran los módulos del backend que se requieren.

4.3 Módulo de usuarios

El módulo de usuarios se encarga de realizar las acciones correspondientes a la gestión de los inicios de sesión y a los registros de los usuarios. Está compuesto por cuatro controladores en la parte del servidor:

- users.authentication.server.controller: es el encargado de realizar el registro de usuarios y guardar los datos recogidos a través del formulario de html en el modelo de usuarios.
- users.authorization.server.controller: define los permisos de usuario y permite al resto de la aplicación saber si un usuario tiene autorización para acceder a determinadas funciones.
- users.password.server.controller: implementa las funciones que permiten cambiar, crear o gestionar las contraseñas. Utiliza la función *crypto.pbkdf2Sync* para generar un hash de manera que las contraseñas sean almacenadas de una manera más segura.
- users.profile.server.controller: es el encargado de actualizar el perfil de usuario y de devolver el objeto JSON del usuario actual.

En el *frontend* existen tres controladores en el módulo de usuarios:

- users.authorization.client.controller: realiza las peticiones HTTP al servidor a las rutas correspondientes en función de si el usuario está intentando registrarse o iniciar sesión y de si la operación ha sido realizada con éxito o no.

- users.password.client.controller: envía los parámetros de las vistas que hacen referencia al reinicio de contraseña usando peticiones post al servidor.
- users.settings.client.controller: se encarga de conectar y eliminar redes sociales del usuario, de manera que pueda elegir con cual de ellas quiere iniciar sesión. Actualmente sólo está operativo el registro a través de Facebook puesto que para el resto de estrategias es necesario una URL operativa y esto aún no es posible puesto que el servidor no está desplegado en un dominio público sino de manera local. En el momento del despliegue de la aplicación web sólo habría que indicar en las estrategias los diferentes ID de aplicación obtenidos en cada una de las redes sociales.

Toda esta lógica de aplicación se ve reflejada a través de las vistas, permitiendo al usuario acceder y modificar la información referente a su perfil a través de formularios web, como se puede observar en la figura **4.2.**

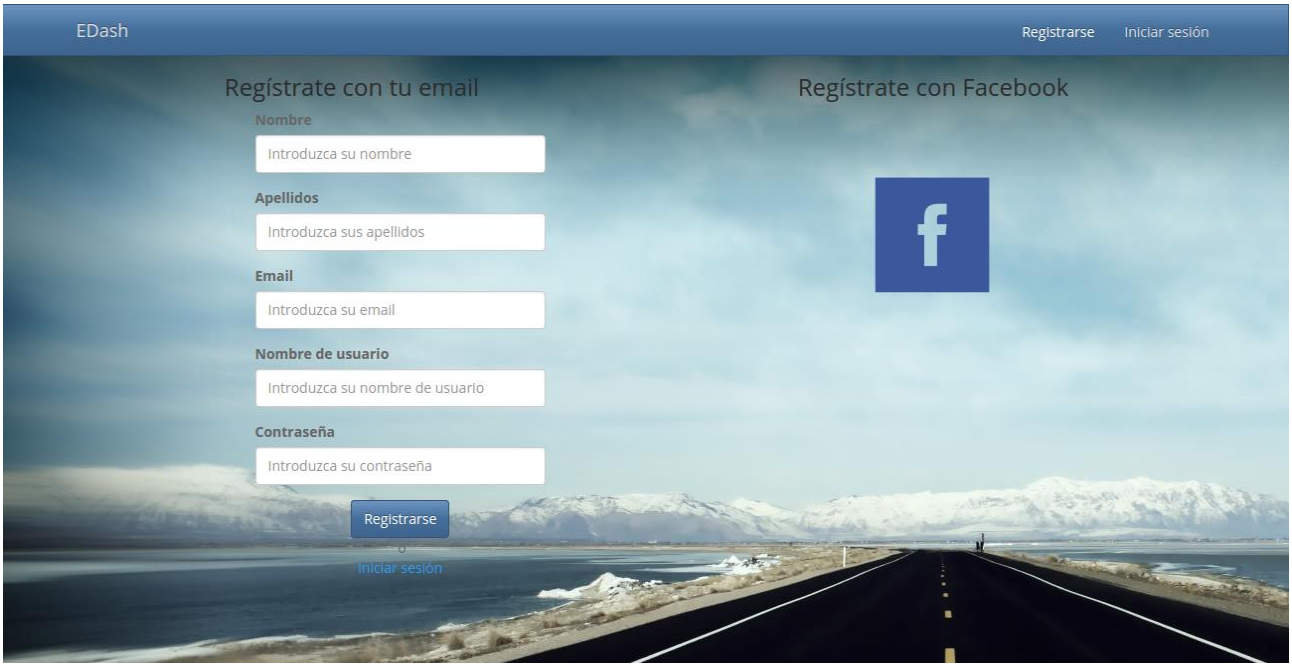


Figura 4.2. Registro de usuarios

4.4 Módulo colisión

En el módulo colisiones se recogen los datos recibidos de los smartphones. En el cuerpo de la solicitud HTTP se encuentran los datos necesarios. Accediendo a ellos, es posible guardarlos en la base de datos. Por ello, fue necesario añadir al modelo los siguientes atributos: fecha del accidente, nombre del vídeo, latitud, longitud, id, email y nombre de usuario.

Una vez establecido el modelo se procedió a la definición de las rutas, que son las encargadas de establecer la comunicación entre las vistas y los controladores. A continuación se listan las rutas que hacen referencia al controlador de este módulo, dividas en dos grupos principales dependiendo de si necesita el paso de un parámetro variable en la ruta o no:

```
app.route('/colisions')
    .get(colisions.list)
    .post(colisions.create);

app.route('/colisions/:colisionId')
    .get(colisions.read)
    .put(colisions.update)
    .delete(colisions.delete);
```

De este modo, las peticiones POST que tiene lugar desde la aplicación se traducen en una llamada a la acción *create* del controlador *colisions*. Este método crea en primer lugar un objeto Colision que cumple con las restricciones del modelo, almacenando los datos enviados a través del servicio:

```
var colision = new Colision({
    video_name: "/modules/colisions/videos/" + req.body.video_name,
    latitud: req.body.latitud,
```

```
    user_id: req.body.user_id,
    user_email: req.body.user_email,
    user_name: req.body.user_name,
    longitud: req.body.longitud
});
```

Después de crear la colisión, se procede a almacenar el contenido del video en un fichero para poder mostrarlo posteriormente:

```
var original_data = req.body.video;
var video = new Buffer(original_data, 'base64');
fs.writeFile('./public/modules/colisions/videos/' + req.body.video_name, video);
```

El último paso consiste en guardar los valores obtenidos y devolver la respuesta a los dispositivos móviles. En caso de que la transacción haya sido realizada con éxito se envía un JSON simple que contiene el string “ok” y en caso contrario se envía un código de error.

```
    colision.save(function(err) {
        if (err) {
            console.log(err);
            return res.status(400).send({
                message: errorHandler.getErrorMessage(err)
            });
        } else {
            res.jsonp(resOk);
        }
```

```
});
```

El siguiente paso a seguir para continuar con el desempeño es implementar las vistas que permitirán a los trabajadores de los servicios de emergencia visualizar las colisiones.

Para la vista general se ha utilizado el módulo de angular *ng-table* que contiene un grupo de directivas para presentar resultados en forma de tabla. De esta manera, en la figura **4.3** se observa la página en la que se muestra el listado de colisiones. El usuario tiene la posibilidad de ordenar las colisiones por el campo que crea conveniente.

Figura 4.3. Listado de colisiones

Los resultados mostrados están paginados para facilitar la visualización y los botones situados en parte inferior de la tabla permiten decidir el número de colisiones mostradas.

Una vez que el usuario hace click en el botón de visualizar colisión, el controlador redirige a la página de ese accidente mostrando como resultado la figura **4.4**. Esta vista muestra todos los datos recogidos durante el accidente. Además, incluye un mapa usando *ng-maps*. Este mapa accede a las coordenadas almacenadas en el modelo y establece un marcador en esa posición.

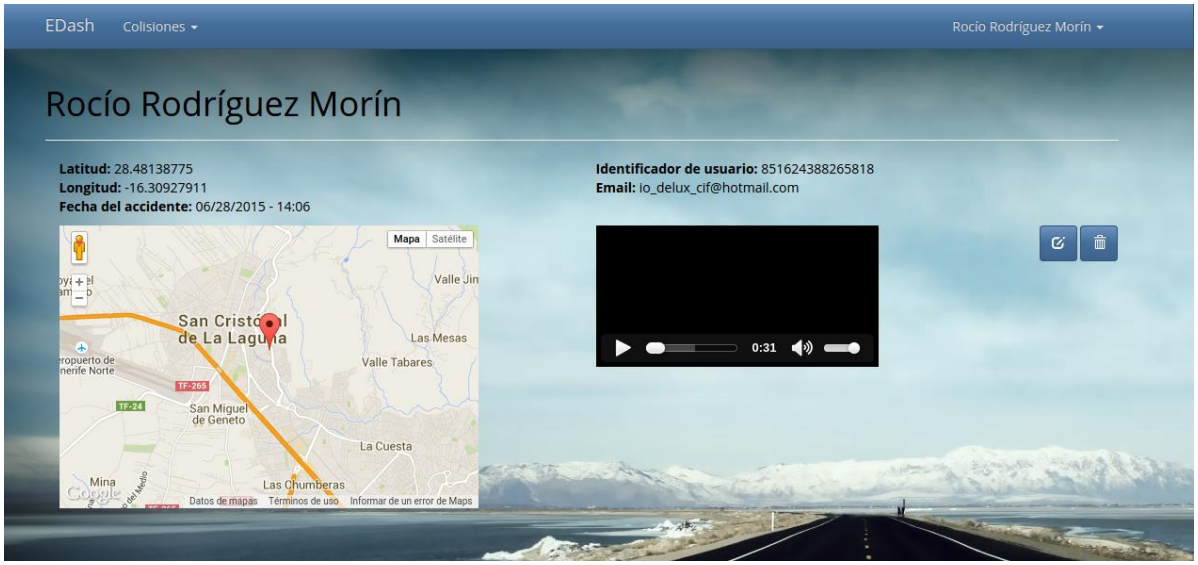


Figura 4.4. Vista de una colisión

A la derecha del mapa podemos ver el vídeo correspondiente al accidente. Además, un botón permite eliminar la colisión en caso de que el trabajador considere que no ha habido colisión.

Capítulo 5. Seguridad

5.1 OpenSSL

La información que maneja E-Dash trata datos de carácter personal que están recogidos en la ley de protección de datos. Además estos datos deben viajar a través de un canal de comunicación que no es seguro, Internet. Esta es la razón por la cual se hace absolutamente necesario implementar un mecanismo de seguridad que permita a ambos extremos saber de quien procede el mensaje y que éste no será leído por terceros. Es decir, hay que garantizar la autenticidad y la confidencialidad de los mensajes usando criptografía [12].

OpenSSL [14] [17] es un proyecto de código abierto que consiste en la implementación de una librería criptográfica y un grupo de herramientas para los protocolos SSL/TLS.

Esta librería fue usada durante el desarrollo del proyecto para generar las claves de AES que se comparten a través del protocolo de intercambio de claves Diffie-Hellman.

En particular, el protocolo de Diffie-Hellman [18] se ha implementado usando criptografía de curvas elípticas, lo que permite utilización de claves de menor longitud con una seguridad equivalente.

En el conjunto de algoritmos que OpenSSL ofrece, se encuentran los citados anteriormente con la versión de AES de 256 bits de longitud de clave. No obstante, la implementación de Diffie-Hellman elíptico ha requerido que este proyecto utilice una versión de la librería superior a la v1.0.0, ya que la suite ECDH no se encuentra en las versiones anteriores.

Una vez que hemos seleccionado los parámetros necesarios, se generan en ambos extremos el par de claves, para proceder al posterior intercambio.

5.2 Diffie-Hellman elíptico

Es un protocolo anónimo de intercambio de claves entre dos partes, cada parte tiene un par de clave elíptica publica-privada, para establecer un secreto compartido por un canal inseguro. Este secreto compartido puede ser usado directamente como clave o como derivado de otra clave que puede ser utilizado posteriormente para cifrar las comunicaciones venideras utilizando un sistema de cifrado. En este caso, este protocolo será el encargado de generar las claves de AES para el posterior cifrado de los datos.

Este procedimiento consiste en intercambiar las claves públicas de los dos extremos de la comunicación. De esta manera, sólo a través del cálculo del secreto se podrá tener acceso a la información.

La única información que se expone es la clave pública. De esta forma, una tercera parte no puede determinar las claves privadas a menos que resuelva el problema del logaritmo discreto de curvas elípticas, un problema de complejidad exponencial.

5.3 Cifrado AES

El Advanced Encryption Standard (AES), también conocido como Rijndael, es un esquema de cifrado por bloque y uno de los más populares en cuanto a criptografía simétrica se refiere. AES puede ser especificado por una clave que sea múltiplo de 32 bits, con un mínimo de 128 bits y un máximo de 256 bits, que es el utilizado en el desarrollo de este proyecto.

AES opera en una matriz de 4x4 bytes, llamada state. En la etapa inicial aplicamos el denominado *AddRoundKey* donde cada byte del *state* se combina con la clave.

En la etapa intermedia aplicamos cuatro pasos: *SubBytes*, que consiste en realizar una sustitución no lineal donde cada byte es reemplazado con otro de acuerdo a una tabla de búsqueda; *ShiftRows,* en donde transponemos cada fila del *state* de manera cíclica un numero determinado de veces; *MixColumns* que

es una operación de mezclado que combinando los cuatro bytes del *state* usando una transformación lineal; y por último volvemos a aplicar el *AddRoundKey*.

En la etapa final aplicaremos el *SubBytes*, *ShiftRows* y A*ddRoundKey*.

Actualmente, E-Dash realiza el envío de datos en una única dirección. Es decir, los smartphones envían las colisiones al servidor pero el servidor sólo devuelve un código que confirma si se ha producido algún error o todo se ha enviado correctamente. Por este motivo, este proyecto implementa únicamente el cifrado del algoritmo AES en la aplicación móvil y el descifrado en el servidor.

Java posee la clase *Cipher* que permite el acceso a las implementaciones criptográficas de cifrado y descifrado. Esta clase no puede ser instanciada directamente, sino que primero se debe llamar al método *getInstance*:

Cipher c = Cipher.getInstance("algoritmo/modo/padding");

Donde el primer parámetro es un algoritmo criptográfico, el segundo es el nombre de un modo de retroalimentación y el tercero es una denominación de un esquema de padding. En este proyecto se ha creado la instancia con la siguiente transformación:

Cipher.getInstance("AES/ECB/PKCS1PADDING");

Esta instancia será utilizada posteriormente por la función init para inicializar el objeto con el modo *ENCRYPT_MODE* y la clave de AES para cifrar como parámetros. Por último, la función *doFinal* procesa los bits almacenados en el búfer y realiza la transformación con la configuración previa que hemos establecido. Al concluir la ejecución de esta función, los datos ya estarían preparados para ser enviados al servidor.

En el lado del servidor OpenSSL realiza el descifrado utilizando la opción decrypt y la clave privada del servidor junto con el secreto compartido por el dispositivo a través del protocolo de intercambio de claves.

Capítulo 6. Presupuesto

Este capítulo contiene un presupuesto detallado de los costes estimados para poder llevar a cabo el proyecto en un entorno real. Se distinguen tres categorías: los dispositivos físicos, las licencias de software y los gastos asociados al personal.

6.1 Dispositivos físicos

En este aparatado se detallan los costes de cada uno de los dispositivos físicos que son necesarios para el desarrollo del proyecto en un entorno real.

Actividad	Unidades	Coste
Dispositivo móvil (WikoLenny)	1	97
Soporte para el coche	1	4,99
Servidor de desarrollo	1	160
Cuenta de desarrollador Android	1	25
Total		**286,99**

Tabla 8.1. Coste de los dispositivos

El coste total de los dispositivos es de doscientos ochenta y seis euros con noventa y nueve.

6.2 Licencias de software

Este proyecto se ha realizado mediante la utilización de herramientas de software libre y, por lo tanto, no existen gastos de licencia de software.

6.3 Personal

En esta sección se recogen los gastos producidos por el personal, estableciendo en la siguiente tabla el número de horas dedicadas a cada una de las actividades. Para el cálculo de los costes total se ha tomado como referencia el sueldo promedio en España de un Ingeniero informático, alrededor de unos 27000 euros anuales para un contrato de jornada completa. Esto establece el precio por hora a 14 euros.

Actividad	Horas	Coste
Investigación y análisis de la tecnología Dashcam	40	560
Estudio de la plataforma de desarrollo Android	75	1.050
Investigación de los algoritmos de seguridad necesarios	50	700
Implementación de la localización	20	280
Implementación de la grabación de vídeo	120	1.680
Implementación de la seguridad	40	560
Implementación del servidor	150	2.100
Cotizaciones de la seguridad social	30%	2.079
Total		**9.009**

Tabla 8.2. Costes de personal

Los gastos derivados de la contratación del personal para la implementación del proyecto ascienden a un total de nueve mil nueve euros.

6.4 Coste total

El coste total de la aplicación del proyecto a un entorno real

Categoría	Coste
Dispositivos	286.99
Costes de personal	9.009
Total	**9295,99**

Tabla 8.3. Coste total del proyecto

En resumen, el coste total del proyecto es de un total de nueve mil doscientos noventa y cinco euros con noventa y nueve céntimos.

Capítulo 7. Conclusiones y líneas futuras

El desarrollo de este proyecto ha permitido llevar a cabo la implementación de una plataforma para servicios de emergencia, consistente en un servidor y una aplicación móvil con el objetivo de aportar como funcionalidad una mejora substancial de los dispositivos llamados dashcams, permitiendo el envío de los vídeos de los accidentes al servidor de forma automática y casi instantánea.

Para el correcto desempeño de este trabajo, ha sido necesario conocer en profundidad el funcionamiento de estos dispositivos y las diferentes tecnologías que intervienen tanto en la aplicación móvil como en el servidor.

Además, en la fase de desarrollo, las implementaciones tanto del servidor como de la aplicación Android se desarrollaron en paralelo para poder realizar las pruebas oportunas a medida que se implementaban cada una de las funcionalidades.

Este proyecto se caracteriza por el gran impacto social que puede llegar a tener. En primer lugar, si se desarrolla en el ámbito de la seguridad vial, puede tener una gran importancia porque facilitaría la asistencia médica en casos de accidente. En segundo lugar, en un futuro se podrían implementar diversas funcionalidades para publicar las imágenes en diferentes redes sociales, lo que implicaría una gran difusión ya que es una tendencia bastante popular en otros países.

Además de estas dos opciones, para implantar el proyecto de forma adecuada sería necesario realizar tests en vehículos reales para definir de una manera más precisa los valores establecidos para la detección de colisiones a través del acelerómetro. De esta manera, la aplicación sería más efectiva ya que enviaría menos falsos positivos al servidor.

Como añadidos se podría implementar diferentes funcionalidades en la aplicación que permitan configurar la calidad de las grabaciones y el espacio límite dedicado a las mismas.

Por último, sería interesante implementar una funcionalidad de la aplicación móvil de manera que el dispositivo se comunique con el usuario por medio de voz, ya que así se producirían menos distracciones y resultaría más cómodo y seguro para los usuarios.

Capítulo 8. Conclusions and open problems

The development of this project enabled to carry out the implementation of a platform for emergency services, consisting of a server and a mobile application in order to provide a substantial improvement and functionality of devices called dashcams, allowing sending videos accidents server automatically and almost instantly.

For the proper performance of this work, it was necessary to know in depth the operation of these devices and the different technologies involved in both the mobile application and the server.

Furthermore, in the development phase, both implementations of server and Android application were developed in parallel to perform the appropriate tests as each of the features is implemented.

This project could have a great social impact. On the one hand, in the field of road safety it could have a great importance because it would facilitate medical care in case of vehicle accident. On the other hand, in the next future it could be possible to implement the functionality to upload videos on different social networks, which is a very popular trend in other countries.

Besides these two options, to implement the project adequately it would be required to do tests on actual vehicles to establish in a more precise way the values set for collision detection through accelerometer. Thus, the application would be more effective, and fewer false positives would be sent to the server.

Furthermore, different functionalities could be implemented in the application to enable the setting of quality of the recordings and the limit space dedicated to them.

Finally, it would be interesting to implement a functionality to enable the communication via voice between the application and the user because in this

way fewer distractions would occur so it would be more convenient and safe for users.

Bibliografía

[1] Shawn Van Every. Pro Android Media: Developing Graphics, Music, Video, and Rich Media Apps for Smartphones and Tablets. Springer. 2009.

[2] Mean stack documentation. http://learn.mean.io/.

[3] Erik Hellman. Android programming: Pushing the Limits. John Wiley & Sons. 2013.

[4] Facebook SDK documentation. https://developers.facebook.com/docs.

[5] Android developers API guides. http://developer.android.com/guide.

[6] Android SDK. http://developer.android.com/sdk.

[7] Darrel Hankerson, Scott Vanstone, Alfred Menezes. Guide to Elliptic Curve Cryptography. Springer Science & Business Media. 2006.

[8] MongoDB. Agility, scalability, performance. https://www.mongodb.org/.

[9] Express, web framework for node.js. http://expressjs.com/.

[10] AngularJS, HTML enhanced for web apps. https://angularjs.org/.

[11] NodeJS documentation. https://nodejs.org/documentation/.

[12] Pino Caballero Gil. Introducción a la Criptografía. Ra-Ma. 2002.

[13] NPM, a package manager. https://www.npmjs.com/.

[14] OpenSSL project. https://www.openssl.org/.

[15] Google console, using google APIs. https://code.google.com/apis/console.

[16] AEPD web page. http://www.agpd.es/portalwebAGPD/index-ides-idphp.php.

[17] OpenSSL source package in Precise. Ubuntu. Retrieved 3 July 2014.

[18] Wikibooks, Cryptography/Meet In The Middle Attack. Retrieved 31 March 2014

[19] Stefan Lucks. Attacking Triple Encryption. Fast Software Encryption, pp. 239-253. Springer. 1998.